(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 August 2001 (09.08.2001)

PCT

(10) International Publication Number
WO 01/57613 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US01/03087

(22) International Filing Date: 31 January 2001 (31.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/496,361 1 February 2000 (01.02.2000) US

(71) Applicant: **VENTRO CORPORATION** [US/US]; 1500 Plymouth Street, Mountain View, CA 94043 (US).

(72) Inventors: **MACKAY, Robert, R.**; 403 East Cattail Court, Saratoga Springs, UT 84043 (US). **TRUJILLO, Kristopher, O.**; 576 North 2310 West, Provo, UT 84601 (US). **BLOYER, Donald, R.**; 1054 Lincoln Lane, Park City, UT 84098 (US). **BERTOLA, Benjamin, K.**; 2085 Pinnacle Terrace Way #306, Salt Lake City, UT 84121 (US).

(74) Agent: **OGILVIE, John, W., L.**; Computer Law++, 1211 East Yale Avenue, Salt Lake City, UT 84105 (US).

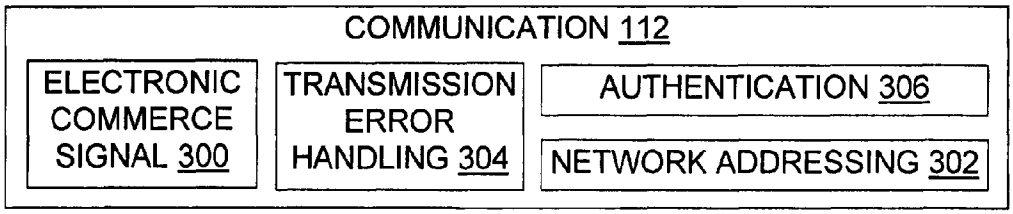
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: HUB AND SPOKE ARCHITECTURE AND METHODS FOR ELECTRONIC COMMERCE



(57) Abstract: The present invention provides tools and techniques for electronic commerce. In a hub and spoke architecture (100), communications (112) are exchanged using XML or another self-documenting format. The communications include electronic commerce transaction data (400) such as purchase order data. Communications may include processing instructions (406) which refer to additional instructions stored at the hub (110). Communications may include a substantially complete history (408) of the transaction's life thus far within the architecture. The invention does not require that each client (102) know detailed information about each other client for commercial communications to be routed. For instance, the hub may select vendors in cases where the client placing a purchase order has specified items and item counts but has not expressly specified or limited the choice of vendor.

WO 01/57613 A2

HUB & SPOKE ARCHITECTURE AND METHODS FOR ELECTRONIC COMMERCE

FIELD OF THE INVENTION

The present invention relates generally to electronic commerce, and relates more particularly to tools and techniques for using a central service provider to perform business transactions between other entities by transmitting transaction data between the entities in a shared self-documenting format.

TECHNICAL BACKGROUND OF THE INVENTION

Electronic commerce could be defined very generally, so that it includes any exchange of business and/or financial information involving any analog and/or digital electronic device. However, the electronic commerce of interest here involves digital computers which are connected by one or more computer networks. Moreover, we are concerned with electronic communications that are directed mainly at commerce between businesses, rather than commerce between individual consumers and a business.

For instance, some businesses use electronic data interchange ("EDI") tools or techniques to exchange commercial information and perform business transactions. Examples of EDI and its uses may be found in United States Patent No. 5,202,977 issued to Pasetes, Jr. et al. ("Pasetes"), United States Patent No. 5,557,780 issued to Edwards et al., United States Patent No. 5,812,669 issued to Jenkins et al. ("Jenkins"), United States Patent No. 5,878,419 issued to Carter, and other public documents. EDI is generally used in the context of a direct physical data transmission connection between two trading partners, as discussed in Pasetes, but Jenkins states that EDI can also be used securely over an open network such as the Internet.

While EDI can provide speed and reliability advantages over corresponding exchanges of paper, EDI also has significant disadvantages as presently used. For instance, EDI documents are written in a complex language agreed upon by their sender and recipient. As a result, numerous cross-references and the need to perform lookup operations complicate the use of EDI, and so does the need for all parties to agree on and implement the specific EDI format to be used in a particular transaction. Dozens or even hundreds of individual EDI formats may be defined for use by a given pair of trading partners.

In response to the cost, complexity, and other disadvantages of relying primarily on EDI for business-to-business commerce, alternatives are emerging. Some alternatives

combine EDI with use of an extensible markup language ("XML"). Unlike EDI, XML is self-documenting in that XML allows users to define information about data structures and content inside an XML document, in a standard way. Because XML has roots in the world wide web's HyperText Transfer Protocol ("HTTP") and Standard Generalized Markup Language ("SGML"), some in the industry believe that XML is also better suited than EDI for standard use on the Internet. XML, EDI, and electronic commerce are discussed further, for instance, in a Harbinger Corporation white paper entitled "eXtensible Markup Language" available through the web site www.harbinger.com. The Harbinger white paper, the patents identified above, and other references are also being filed by applicants with the U.S. Patent and Trademark Office to be made of record in this application.

In short, a wide variety of approaches are being proposed and/or used to facilitate business-to-business electronic commerce. In some cases, the relative strengths and weaknesses of the approaches are well understood; in other cases, firm conclusions are harder to reach. In this type of dynamic environment, which is filled with both risk and promise, it would be an advancement in the art to provide new tools and techniques for facilitating electronic commerce. It would be a further advance to do so without abandoning in every instance the older components of such commerce, such as EDI and the so-called "legacy systems" used within a given business entity. Accordingly, new tools and techniques which increase the choices available for performing electronic commerce are disclosed and claimed herein.

BRIEF SUMMARY OF THE INVENTION

The present invention provides tools and techniques for electronic commerce. In particular, some embodiments of the invention include a hub & spoke architecture for integrating disparate systems into transaction communities with migrating encapsulated process information. The architecture is a hub & spoke architecture (alternately denoted a "hub-and-spoke" or "hub and spoke" architecture) in that one or more servers at one or more logically central locations service client programs at various client business entities; the server is at the hub, and the client entities define outlying spokes extending from the hub. The logically central location is central in the network topology sense; it is not necessarily physically or geographically central. This hub & spoke topology is in contrast with a peer-to-peer topology, for instance, in which clients would communicate directly with each other instead of going through a central location.

The hub is also central in a business role sense, because transactions between clients are assisted by the hub. For instance, the hub monitors activity to identify purchase orders that have been sent but not filled, and takes follow-up action accordingly. In some embodiments, the hub includes procurement software which chooses the supplier(s) best suited for filling a given purchase order. That is, the buyer need not specify particular suppliers, although that can be done; it is enough to specify the desired items and their respective quantities. As a result of these hub capabilities, the hub is more than a mere conduit for data. The hub does provide data transmission or networking services, but the hub is also an active provider of electronic commerce services such as transaction monitoring and procurement management services.

The various client entities may use different electronic commerce programs internally. For example, one client entity might use a particular set of EDI formats, while another client entity uses a different set of EDI formats, a third entity uses custom procurement software and a relational database, and a fourth entity uses a combination of spreadsheets and paper forms. The inventive clients can translate these and other disparate commercial documents into a shared self-documenting format which is used in the transmissions between the hub server and the clients involved in a given transaction. Thus, the invention integrates disparate client entities into a community for the duration of the transaction in question.

Unlike approaches that maintain log files at the network nodes in an electronic commerce network, the present invention maintains migrating encapsulated process information. That is, the hub server and clients maintain within the transmissions for a given transaction a substantially complete history of that transaction. This allows the hub and the clients to detect, diagnose, and at least attempt to overcome problems that may arise, without requiring synchronization or replication of logs. Separate logs may also be used, but they are not required, because the log is maintained in the transmissions for the transaction.

A complete history of the steps taken to complete the transaction qualifies as a substantially complete history, but an entirely complete history is not always required. Data and/or events may be omitted from the history if they are not needed to address unacceptable delays, lost transmissions, scrambled transmissions, payment refusal, or other electronic commerce problems. Thus, the invention only requires a history which is "substantially" complete, not one that contains every detail of every event involving the transaction at hand.

In addition to an architecture for electronic commerce, the invention provides electronic commerce signals, methods, and configured storage media. In each case, the

invention provides new tools and techniques for facilitating electronic commerce without necessarily abandoning use of EDI, enterprise resource planning software, other procurement software, spreadsheets, relational databases, and so on within the client entities. XML, commercial XML, or another self-documenting language is used for communications between clients and the hub. Familiar tools for encryption, remote software updates, and data transmission in a network are readily adapted for use according to the present invention. Other features and advantages of the present invention will become more fully apparent through the following description.

BRIEF DESCRIPTION OF THE DRAWINGS

To illustrate the manner in which the advantages and features of the invention are obtained, a more particular description of the invention will be given with reference to the attached drawings. These drawings only illustrate selected aspects of the invention and thus do not limit the invention's scope. In the drawings:

Figure 1 is a diagram illustrating a hub & spoke architecture for electronic commerce according to the present invention.

Figure 2 is a diagram further illustrating a client according to the present invention, with reference to the clients shown in Figure 1.

Figure 3 is a diagram further illustrating an electronic commerce communication according to the present invention, with reference to the communications shown in Figure 1.

Figure 4 is a diagram further illustrating an electronic commerce signal of the present invention, from a communication of the kind shown in Figures 1 and 3.

Figure 5 is a diagram further illustrating data flow during an example use of the hub & spoke architecture shown in Figure 1.

Figure 6 is a flow chart further illustrating methods of the present invention.

Figure 7 is a diagram illustrating a hub & spoke architecture for electronic commerce in use by a hospital according to the present invention.

Figure 8 is a diagram further illustrating data processing in a client according to the present invention.

Figure 9 is a diagram further illustrating steps and structures in a client according to the present invention.

Figure 10 is a flow chart further illustrating processing at the hub according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention provides business tools and techniques for facilitating electronic commerce. The invention may be embodied in various ways, and it may also be described in various ways. Accordingly, the examples discussed throughout this document are not necessarily consistent with one another in every particular; rather than omitting details to achieve complete consistency, details are included if they may assist in a better understanding of an embodiment of the invention.

Overview

The invention provides a network-based design for electronic commerce, namely, a design having novel features at a transactional level. In other words, the invention provides a commerce network built on top of one or more otherwise conventional computer networks. A "transaction" in the commerce network is viewed as a business transaction, as opposed to a mere exchange of data packets or some other network operation at the network physical level. Client entities who participate in electronic commerce using the commerce network do not require specific computer networking knowledge of each other. For instance, they do not need to know the other participant's computer system information, IP or other network delivery addresses, physical location, or any other data needed to transmit communications electronically; these details are handled by the hub server and the commerce client software.

The invention utilizes conventional computer networking technology with novel additional tools and techniques in order to let client entities execute business transactions in an electronic form. For instance, the invention permits use of virtually any available network communication protocol(s), such as Ethernet, SLIP, Token-Ring, TCP, and so on. Likewise, the invention permits use of virtually any available network transport mechanism(s), such as HTTP, SMTP, FTP, and so on.

One system according to the invention is capable of translating commercial data between the data formats that are used by client entities and a common transactional language format that is shared by all clients in the commerce network. The invention transports business data from one client entity to another client entity through a central hub that is operated by an electronic commerce service provider. Each client entity has a separately installed client software program (called the "client") for communications with the central

hub. This defines a hub & spoke topology for the commerce network; the underlying computer networks may have ring, web, hub & spoke, or one or more other topologies.

By contrast with many EDI systems or another direct connection approach, which require specific physical interconnections between client entities, the commerce network is created without such specific physical interconnections because it uses Internet or world wide web networks. Some current EDI systems also use the Internet for data exchange. Even in such cases, however, it is necessary for one peer to directly identify the other peer with whom data will be exchanged.

By contrast, embodiments of the present invention use the central hub to connect the entities. In particular, a buyer entity may place an order without directly identifying the seller. The hub will select one or more seller(s) to fill the order after reviewing the purchase order to identify desired items, accessing catalogs, price lists, and other familiar components, and checking lists or other structures that identify sellers in the hub & spoke architecture. Thus, the buyer does not need to specify in each instance the address of the target vendor entity.

The communications across a variety of client entity platforms is standardized within the hub & spoke architecture by use of a self-describing data structure, such as one implemented with XML. A given inventive communication transmitted between client and hub typically contains business information, electronic commerce routing instructions (as opposed to computer network router or bridge instructions), and electronic commerce processing instructions (such as who should be notified at the client entity when the communication arrives). Suitable business information includes data for an electronic commerce transaction, such as purchase order data, invoice data, catalog data, and so on. By using XML, this data can be easily described in the communication, unlike similar data in EDI exchanges, for instance. The descriptions of transaction data are preferably set forth in a consistent fashion by using a single internal (hub-client) format. This is different from electronic commerce systems in which XML is used to implement a variety of definitions for compatibility with still other systems, thereby increasing the system complexity and making it more difficult to maintain and support.

Another notable feature of the present communications is that they typically each contain a substantially complete transaction history. The history is located in the same self-describing structure, and plays a significant role in supporting the electronic commerce infrastructure. For instance, the history supports transaction management in the loosely coupled commerce network. Proactive management of transactions is important to the

ongoing success and performance of an inventive system, as are somewhat similar steps in other connected disparate entities. For instance, in a computer network routers and switches will support recasting or rebroadcasting of packets that are unconfirmed. This is an immediate response to failure. In the inventive commerce network, retransmissions may also be called for, but there is normally a longer delay between action and reaction than there is at the network packet level.

Transaction lifecycle and the steps involved in executing a transaction are tied directly to the document data by means of the embedded transaction history, which is carried with the document at each of the various points of system interaction. Through this method of local history support, any commercial document in the system can be tracked, and problems can be resolved without centralized administration.

For instance, assume the problem is a failed purchase order acknowledgment. That is, the hub expected to receive from a vendor client an acknowledgment of a purchase order communication forwarded to the vendor client by the hub from a buyer client. The transaction failure is discovered and identified by reviewing the history in the transaction and measuring the amount of time since the last history event. Measurement may use rules embedded in the communication to determine if the elapsed time is too long. Other embedded rules cause the hub to send a request for response and/or to notify a particular person of the problem by email or pager.

Without the invention, it might be necessary to contact the buyer client to determine the time the order was placed by their system, contact their account manager to determine the acceptable delay, calculate the delta between these times, compare it to the elapsed time, and then contact the buyer client again to ask that a confirmation of the order be sent. This example treats a very simple problem. As problem complexity escalates, so does the level of effort required to resolve the problem, and the corresponding benefits of automation with the invention.

The novel system has been designed to allow abstraction from the delivery type (e.g., file transfer, message transfer, database transfer, electronic mail transfer), the transaction data (e.g., purchase order, catalog), and the delivery mechanism (e.g., FTP for files, JMS for messages) used in a particular instance. The invention can be used in virtually any computing environment currently in production, as long as that environment includes at least a Java Virtual Machine or its approximate equivalent, and a network connection of some form.

Encryption in the inventive system may be based on a variety of solutions. Encryption is often related to the transport type, since differing transportation types use differing mechanisms for security. Transaction data may also be encrypted by the client independently of and in addition to any encryption performed by the delivery mechanism.

The user interface for the system is based upon a centralized website that provides support for various system functions, such as: configuration capabilities, including setting the network addresses for delivery, notification, monitoring and reporting; output mapping and translation instructions; delivery notification; receipt confirmation; software upgrades to clients; and data requests requiring selection and incremental delivery of catalog data, for example. The centralized web site (also referred to as the "hub" or "hub server") thus supports the remote client installations. In some embodiments, the hub can reproduce the client installation exactly after a total client loss at the remote system.

The centralized site also provides customization capabilities for the output formats provided by the client in the form of mappings that allow an XML programmer, for instance, to implement translations between specific client entity formats and the shared self-documenting format of the hub-client communications. The mappings are embedded in the communications, so they are generally available with each delivery of a transaction to a remote client. The resulting flexibility allows for a dynamic change to an output format from a centralized location that is then executed with the next transactional delivery. The change in mappings between two transactions does not cause either transaction to fail.

The central site also allows for synchronized software module addition, whereby each client is kept in sync with the server and each other client that is also using a given module. For instance, a remote client or the hub may add a new format adapter. An adapter addition at the client generates a communication that includes the added module, the configuration file, and contact information that is delivered to the server and posted as a client change. Similarly, the server may post a client change to the configuration at the server, which would then be reflected to the client to be decrypted if necessary, decompressed if necessary, otherwise extracted from the communication, and installed at the client entity.

Accordingly, embodiments of the invention may include a transaction processing monitor ("TPM") at the hub which supports a loosely coupled network of remote nodes (clients) for the purpose of commercial data exchange. Clients do not necessarily perform transaction processing monitoring, although the system does support internal transactional integrity. Although the inventive system enforces transactional integrity through the

confirmation of delivery of a given step in a transaction, it is an EDI/asynchronous-style enforcement process, not a TPM-style transaction. Some embodiments provide rollback capabilities at the client as well as the hub. In a sense, the invention thus combines the high-response transactional model of a TPM with the loosely-coupled transactional model of EDI.

It may also be stated that, rather than being request based system, the inventive system is a workflow driven, event driven system in which the server and client each push data without a request to each other. In other words, a request model need not be involved. The only time a response is generated is to confirm the receipt of the push.

Architecture

Figure 1 illustrates a hub & spoke architecture according to the present invention, indicated generally at 100. The architecture 100 facilitates electronic commerce between client entities 102 by providing services managed by a service provider 104. Electronic commerce may also be performed between a client entity 102 and the service provider 104 using the architecture 100.

Each client entity 102 will have some type of transaction system 106. The systems 106 may be installed in connection with the corresponding client entity's subscription to the service provider 104 services, but most systems 106 will already be present when the client 108 is installed. Likewise, a given system 106 will generally be used only internally within a particular client entity 102, but in some cases the system 106 may be outsourced and/or located at a data center leased by the client entity 102. To distinguish these systems 106 from the other elements shown, the systems 106 are termed "single-entity" systems; they are sometimes referred to elsewhere as "legacy systems". Unlike client software and/or hardware 108, and unlike hub server software and/or hardware 110, the systems 106 do not normally link trading partners to allow communications 112 as taught herein. However, the fact that a system 106 may support EDI or another conventional electronic commerce capability does not prevent it from being a "single-entity" system with respect to the present invention.

Clients 108 are novel elements of the present invention, provided specifically to support communications within the hub & spoke architecture 100. Clients 108 may be implemented in some embodiments as software installed on general-purpose computer hardware, and in other embodiments may include special-purpose hardware according to the present invention. The line between hardware and software becomes increasingly indistinct as

tools like application-specific integrated circuits ("ASICs"), field programmable gate arrays ("FPGAs"), and so on become more widely available.

In operation, clients 108 perform as discussed in the Overview and Development History sections, and elsewhere herein. For instance, in one embodiment a client 108 at a buyer entity receives purchase order data from the single-entity transaction system 106, translates it from the system 106 format (e.g., EDI) into a self-documenting format such as an XML format, creates a communication 112 containing the data and related information such as the history of the transaction thus far, and sends the communication to the hub server 110. A client 108 at a vendor entity receives the XML data, extracts the purchase order data, optionally sends the hub server an acknowledgement confirming receipt of the purchase order communication 112, converts from XML to the client entity system 106 format, and submits the purchase order data in that format to the vendor's system 106. Clients operate similarly with regard to invoices, catalog request and fulfillment, and other commercial transactions. Communications 112 are discussed further in connection with Figures 3 and 4, and elsewhere herein.

A client 108 may exchange data with the local single-entity transaction system 106 by way of a buffer 114 using a post-and-subscribe approach, semaphores, signals, or other familiar inter-process communication tools and techniques. The buffer 114 includes volatile memory such as RAM and/or non-volatile memory such as a magnetic hard disk. Clients 108 and their interfaces with the systems 106 and the hub server 110 are discussed further elsewhere herein, such as in connection with Figure 2.

In its various embodiments, the hub server 110 generally supports asynchronous message storage and retrieval, through message queues or similar mechanism. Suitable implementations may use, for instance, IBM MQSeries software, TIB/Rendezvous software, Weblogic App Server software, or other message-based middleware and/or Java application server software in a messaging and information infrastructure.

Like the clients 108, the server 110 includes code that understands the format of communications 112 to embed or extract transaction data, provides network routing, and provides confirmation upon receipt of a message 112. Extendable delivery mechanisms provide for actual transmission of the message 112 through the network "wire". These work with a routing system that contains client 102 addresses, and a centralized store containing all the configuration details, such as which transport adapters are installed at which clients.

The hub 110 preferably also includes exception handling capability as part of the transaction management logic. When expected messages 112 do not arrive, the server 110 takes appropriate steps. For instance, a notification system ties exception handling and transaction management together and provides email or other notification to designated persons at the involved entities 102, 104.

In some embodiments, the service provider 104 also provides procurement accounting hardware and/or software 116, such as enterprise resource planning ("ERP") software, inventory databases and database managers, and other conventional procurement management tools. The procurement module 116 exchanges data with the hub server 110. When present, the procurement software 116 provides the hub server with access to business logic, possibly including access to internal systems of the service provider 104 which are not directly available to other systems in the environment 100. For instance, the procurement software 116 may identify the best supplier for a given set of items in a purchase order. The procurement software 116 may include and/or access accounting data, transactional data, reporting information, product information, access control information, credit status, and other business data as deemed necessary to support the hub 110 and the commercial transactions requested by clients 108.

Like the other Figures and examples given herein, Figure 1 is illustrative only. In various embodiments of the invention, components shown in Figure 1 may be omitted, repeated, renamed, or grouped differently. For instance, the leftmost client entity 102 in Figure 1 includes a firewall 118 between the client 108 and the external network, including the hub server 110. Other client entities 102 do not necessarily have a firewall, as illustrated by the rightmost upper client entity 102 in Figure 1. In some embodiments all client entities 102 have firewalls, in some none have firewalls, and in some embodiments a client 108 resides outside the firewall rather than inside it. Similarly, Figure 1 shows three client entities 102, but other embodiments may have more or fewer client entities 102. As noted earlier, some embodiments also omit the procurement accounting module 116.

Clients

Figure 2 illustrates components of a client 108. As noted, the client 108 may be implemented as software which configures a general-purpose computer according to the invention, or it may be implemented with special-purpose hardware such as ASICs or

FPGAs. The client 108 may be a virtual user of the systems 106, in that the client 108 (which is a mechanism rather than a person) exchanges data with the systems 106.

The illustrated client 108 includes an incoming data processor 200, an outgoing data processor 202, one or more format adapters 204, and one or more transport adapters 206. As noted in the Development History section, other implementations of the client lack pluggable adapters 204, 206, using "hard-coded" format conversion routines and/or hard-coded transport protocols instead. Either way, the client 108 uses one or more "mappings" to perform format conversion(s). The mappings can be written by any programmer familiar with XML and/or XML Style Language ("XSL"). Some embodiments of the invention embed XSL in the XML file, which is somewhat unusual in the world of XML uses.

The incoming data processor 200 checks for incoming communications 112 from the hub server 110 using at least standard computer networking tools and techniques. The transport adapter 206 or other transport mechanism may be viewed as part of the incoming data processor 200; as with Figure 1, components of Figure 2 may be grouped differently in different embodiments of the invention. After receiving a communication 112, the incoming data processor 200 extracts electronic commerce transaction data from the communication 112, and posts the data to the single-entity-transaction-system 106. The data extraction may be defined to include format conversion from XML to the local system 106 format; once again, we see that embodiments may group illustrated components differently without thereby removing them from the scope of the invention. Posting the data may be performed by copying it to the buffer 114, and setting a flag or otherwise ensuring that the system 106 will recognize that new data has been posted for the system 106 to process.

Suitable format adapters 204 include adapters between the client-hub communication 112 self-documenting format and the format used by a single-entity-transaction-system 106. Familiar component plug-in techniques, including techniques from Java, object-oriented programs, COM, or other sources, may be used to include or exclude particular adapters. Conversion between XML and legacy system 106 formats may be performed using familiar parsing techniques.

Suitable transport adapters 206 include adapters that provide connectivity using familiar protocols, such as HTTP, FTP, ARCNET, Ethernet, Token ring, and so on. Some type of support for Transport Control Protocol ("TCP") connectivity is preferred. Naturally, the hub server 110 includes corresponding transport support for each client 108 in the architecture 100.

The outgoing data processor 202 operates in a manner that is generally similar to that of the incoming data processor 200. The outgoing data processor 202 checks for outgoing electronic commerce transaction data from the single-entity-transaction-system 106 by receiving an interrupt, polling the buffer 114, or another familiar method. The outgoing data processor 202 embeds at least part of the electronic commerce transaction data (some data may be strictly for internal client entity 102 use) in a communication 112 in a self-documenting format, and sends the communication 112 to the hub server 110. The format adapter(s) 204 are thus either part of, or used by, the outgoing data processor 202, depending on whether a given embodiment regroups the illustrated components, or does not regroup, respectively. Likewise, the transport adapter(s) 206 may be viewed as being either part of, or simply used by, the outgoing data processor 202.

Communications

Figure 3 illustrates a communication 112 in greater detail. A business transaction performed with the invention includes one or more communications 112 between hub server 110 and client(s) 108. A communication in turn includes at least a novel electronic commerce signal 300, and conventional computer networking data 302 such as IP addresses and/or Uniform Resource Locators ("URLs"). As discussed in connection with Figure 4 and elsewhere herein, the electronic commerce signal 300 includes transaction data such as a purchase order, as well as a history of the transaction thus far, and possible other components.

A communication 112 is independent of the transport protocol used, and need not specify how it should be transported. However, a communication 112 on the "wire" may include checksums, error detection codes, error correction codes, and/or other familiar means for transmission error handling 304. Such components 304 may be present in the electronic commerce signal 300 and pertain only to the contents of that signal 300, or they may be calculated using different or additional components of the communication 112. As with the other Figures, illustrated components of Figure 3 may be omitted, renamed, repeated, or grouped differently in different embodiments of the invention.

A communication 112 may include digital signatures, tokens, certificates, private or public keys, or other familiar means for authentication 306. Again, a given instance of the communication 112 may include authentication component(s) 306 inside the electronic commerce signal 300, outside that signal 300 but within the communication 112, or both.

Communications 112 and electronic commerce signals 300 may be embodied according to the invention in twisted pair, coaxial, or optical fiber cables, telephone lines, satellites, microwave relays, modulated AC power lines, and/or other data transmission "wires" of the underlying computer network. Communications 112 and electronic commerce signals 300 may also be embodied in computer memory in the architecture 100, including volatile memory such as RAM and non-volatile memory such as a hard disk.

Communications 112 and/or electronic commerce signals 300 may be encrypted (and, of course, decrypted) and/or compressed (and decompressed) using familiar tools and techniques. In particular, communication 112 encryption may be performed by a client 108 regardless of whether the underlying transport mechanism for transmitting packets to the hub 110 also performs encryption.

The following example is provided to further understanding of the communications 112 and the present invention:

```
<TRANSACTION docType="832" customerID="999999" origSysRef="283727343">
      <DELIVERY_METHOD id="21" name="JMS Delivery Method" code="JMS"
description="This delivers by JMS"
jndihomename="promedix.ia.TestAdapterHome">
            <jmsTopicName>promedix.test.TestingTopic</jmsTopicName>
            <jmsTopicURL>t3://peach.promedix.com:7002</jmsTopicURL>
      </DELIVERY_METHOD>
      <CLIENT_DOCUMENT docType="832">
            <ADAPTERS>
                  <ADAPTER classname="com.promedix.adapters.XSLAdapter"
selector="850" regexp=".*order.*|.*850\.$">
                  <ADAPTER_STYLESHEET>
                        <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
                              <xsl:template match="CatalogRequest">
                                    <html>
                                    <body bgcolor="#ffffff">
                                    <center>
                                    <h1>Catalog created on <xsl:value-of
select="@RequestDateTime"/>
                                    </h1>
                                    <p>
                                          <font size="+1">Grouped by
<xsl:value-of select="@GroupedBy"/></font>
                                    </p>
                                    <p>
                                    <table border="1">
                                    <tr>
                                          <td bgcolor="#7fffd4"><b>Supplier
ID</b></td>
                                          <td bgcolor="#7fffd4"><b>Sku
ID</b></td>
                                          <td
bgcolor="#7fffd4"><b>Name</b></td>
                                          <td bgcolor="#7fffd4"><b>Unit of
Measure</b></td>
```

```
                                        <td bgcolor="#7fffd4"><b>Mfg Part
Number</b></td>
                                        <td bgcolor="#7fffd4"><b>List
Price</b></td>
                                  </tr>
                                  <xsl:apply-templates select="Product"/>
                                  </table>
                                  </p>
                                  </center>
                                  </body>
                                  </html>
                            </xsl:template>
                            <xsl:template match="Product">
                                  <tr>
                                        <td><xsl:value-of
select="@SupplierID"/></td>
                                        <td><xsl:value-of
select="@SkuID"/></td>
                                        <td><xsl:value-of
select="@Name"/></td>
                                        <td><xsl:value-of
select="@UnitOfMeasure"/></td>
                                        <td><xsl:value-of
select="@MfgPartNumber"/></td>
                                        <td>$<xsl:value-of
select="@ListPrice"/></td>
                                  </tr>
                            </xsl:template>
                      </xsl:stylesheet>
                </ADAPTER_STYLESHEET>
                </ADAPTER>
          </ADAPTERS>
     </CLIENT_DOCUMENT>
     <TRANSACTION_DOCUMENT>
          <CatalogRequest GroupedBy="Supplier" RequestDateTime="Mon Jan
10 10:07:34 MST 2000">
          <Product MfgPartNumber="stuff12" SupplierID="promedix"
ListPrice="20.25" UnitOfMeasure="ea" Name="stuff12" SkuID="stuff12"/>
          <Product MfgPartNumber="test3" SupplierID="promedix"
ListPrice="0.1" UnitOfMeasure="ea" Name="test3" SkuID="test3"/>
          <Product MfgPartNumber="stuff14" SupplierID="promedix"
ListPrice="20.0" UnitOfMeasure="ea" Name="stuff14" SkuID="stuff14"/>
          <Product MfgPartNumber="stuff15" SupplierID="promedix"
ListPrice="99.99" UnitOfMeasure="ea" Name="stuff15" SkuID="stuff15"/>
          <Product MfgPartNumber="stuff16" SupplierID="promedix"
ListPrice="2.25" UnitOfMeasure="ea" Name="stuff16" SkuID="stuff16"/>
          <Product MfgPartNumber="test1111" SupplierID="promedix"
ListPrice="19.99" UnitOfMeasure="cs" Name="test" SkuID="testy"/>
          <Product MfgPartNumber="promedix111" SupplierID="promedix"
ListPrice="19.99" UnitOfMeasure="cs" Name="product for promedix.com"
SkuID="promedix111"/>
          </CatalogRequest>
     </TRANSACTION_DOCUMENT>
     <HISTORY_DOCUMENT>
          <History>
               <Caller name="Client">
          <Entry message="Imported order from host erp" datetime="2000-
01-17" status="Success"/>
          <Entry message="Set order to server for processing"
datetime="2000-01-17" status="Success"/>
```

```
            </Caller>
            <Caller name="ListenerFactory">
            <Entry message="Order pickedup by the ListenerFactory for
processing" datetime="2000-01-17" status="Attempt"/>
            </Caller>
            <Caller name="850Processor">
            <Entry message="Inserted order into the ERP system, run time
was 11 seconds." datetime="2000-01-17" status="Attempt"/>
            </Caller>
            </History>
      </HISTORY_DOCUMENT>
      <CLIENT_CONFIGURATION>
            <CLIENTHEADER>
                  <TRANSPORT classname =
"com.dotconnect.transport.ClientSMTPTransport">

                        <LISTENER classname =
"com.dotconnect.listener.SMTPInboundListener"
                                                      sourceurl =
"/usr/home/rbloyer/tenetdest">
                        </LISTENER>

                        <LISTENER classname =
"com.dotconnect.listener.DirectoryListener"
                                                      sourceurl =
"/usr/home/rbloyer/tenet850src/">
                        </LISTENER>

                  </TRANSPORT>

                  <POOLCONF>

                        <POOL classname =
"com.dotconnect.pool.JMSSessionPool"
                                          canautostart = "true"
                                          cankill = "true"
                                          defaultpoolsize = "5"
                                          inboundtopic =
"promedix.jms.InboundJMS"
                                          outboundtopic =
"promedix.jms.OutboundJMS"
                                          provider_url =
"t3s://smurf:7003"/>

                        <POOL classname = "com.dotconnect.pool.ThreadPool"
                                          usethreadpooling = "false"
                                          defaultpoolsize = "9"
                                          incrementinst = "5"/>

                  </POOLCONF>


                  <REGISTRY>

                        <CLIENT_DOCUMENT type = "850"

      xsldocument = "/usr/home/rbloyer/tenetxsl/850.xsl"

      mapdocument = "/usr/home/rbloyer/tenetmap/850map.xml"
```

```
        conversionclass = "com.dotconnect.conversion.FixedLengthLoader"
        selector = "850"
        regexp = ".*order.*|.*850\.xml$">

                                        <ADAPTER classname =
"com.dotconnect.adapter.SMTPSendAdapter"
                                                version = "1.0"
                                                date = ""
                                                expdate = ""
                                                partner = ""
                                                author = ""
                                                isvalid = ""
                                                opt1 = ""
                                                opt2 = ""/>

                        </CLIENT_DOCUMENT>

                        <CLIENT_DOCUMENT type = "855"
        xsldocument = "/usr/home/rbloyer/tenetxsl/855.xsl"
        mapdocument = "/usr/home/rbloyer/tenetmap/855map.xml"
        conversionclass = "com.dotconnect.conversion.FixedLengthLoader"
        selector = "855"
        regexp = ".*ack.*|.*855\.xml$"
        sourceurl = "/usr/home/rbloyer/tenetdest">
        processInstruct = "/usr/batch/load855.sc"

                                        <ADAPTER classname =
"com.dotconnect.adapter.DefaultReceiveAdapter"
                                                version = "1.0"
                                                date = ""
                                                expdate = ""
                                                partner = ""
                                                author = ""
                                                isvalid = ""
                                                opt1 = ""
                                                      opt2 = ""/>

                        </CLIENT_DOCUMENT>

                        <CLIENT_DOCUMENT type = "832"
        xsldocument = "832.xsl"
        mapdocument = "/usr/home/rbloyer/tenetmap/832map.xml"
        conversionclass = "com.dotconnect.conversion.FixedLengthLoader"
        selector = "832"
```

```
        regexp = ".*catalog.*|.*832\.xml$"

        sourceurl = "/usr/home/rbloyer/tenetdest">

                                        <ADAPTER classname =
"com.dotconnect.adapter.DefaultReceiveAdapter"
                                                        version = "1.0"
                                                        date = ""
                                                        expdate = ""
                                                        partner = ""
                                                        author = ""
                                                        isvalid = ""
                                                        opt1 = ""
                                                        opt2 = ""/>

                        </CLIENT_DOCUMENT>

                </REGISTRY>

                <PROPERTIES lastmodified = "0">
                        <PARTNER
                                custcode = "999999"
                                companyname = "Empire"
                                contactname = "DarthSidius"
                                contactphone = "way-long-distance"
                                contactemail = "dsidius@deathstar.com"
                                processingbox = "devmail@promedix.com"
                                monitorrecipients = "rbloyer@promedix.com"
                                errorrecipients = "rbloyer@promedix.com"
                                cycletime = "10000"
                                clientversion = "1.0"
                                serialnumber = "123456789"/>

                        <MAILCONFIG
                                cansendmail = "true"
                                smtphost = "sherbet.promedix.com"
                                pophost = "sherbet.promedix.com"
                                imaphost = ""/>

                        <FILES
                                keyfile = "com/dotconnect/config/smtpkey.ser"
                                securefile =
"com/dotconnect/config/promedix_smtp.ser"/>

                        <LOGS
                                enablelogging = "true"
                                clientlog = "client.log"
                                errorlog = "err.log"
                        />

                </PROPERTIES>
        </CLIENTHEADER>
    </CLIENT_CONFIGURATION>
</TRANSACTION>
```

Electronic Commerce Signals

Figure 4 illustrates an electronic commerce signal 300 in greater detail. As with the other Figures, components shown may be omitted, repeated, renamed, and/or grouped differently in different embodiments of the invention, unless called for by the relevant claims.

The illustrated electronic commerce signal 300 includes electronic commerce transaction data 400. Suitable transaction data 400 includes, without limitation, data which partially or completely defines a purchase order, order status inquiry, invoice, payment, catalog request, and/or catalog fulfillment for electronic commerce between separate entities. The data 400 is in a self-documenting format, using a language such as XML.

The illustrated electronic commerce signal 300 also includes a transaction type identifier 402 which identifies the type of commercial transaction to which the signal 300 pertains. This type identifier 402 may be separate from the transaction data 400 as illustrated, or it may be part of the transaction data 400 in other embodiments. Suitable identifications include those corresponding with the transaction data 400 possibilities noted elsewhere herein, such as "purchase order" and "invoice". The transaction type identifier 402 may be implemented using bitflags, ordinal numbers, predefined text strings, or other familiar operation or record type identifier means.

The illustrated electronic commerce signal 300 also optionally includes electronic commerce routing instructions 404. These instructions identify the target trading partner by name or in another manner convenient to non-technical users of the novel system. The hub server 110 translates the non-technical destination identifier 404 into IP addresses, URLs, Ethernet addresses, and/or other computer network addresses using a simple lookup table or other familiar technique adapted for use according to the present invention.

The illustrated electronic commerce signal 300 also optionally includes electronic commerce processing instructions 406. For instance, the instructions 406 may specify email addresses or pager numbers for people to be notified when a purchase order arrives at a vendor client entity 102. The instructions 406 may identify an application 106 at the client entity 102 to process data after the client 108 creates a file in the buffer 114, and call that application 106. The instructions 406 may identify a database instance 114 at the client entity 102 to receive data 400, and insert the data 400 directly into the database.

The illustrated electronic commerce signal 300 also includes a substantially complete history of the commercial transaction thus far 408. For instance, the history 408 specifies the current settings of the remote software 108, the currently installed software adapter(s) 204

and/or 206, and the currently activated transaction type(s) 402. An example is provided in the XML HISTORY_DOCUMENT above.

Example Using a Purchase Order and Invoice

To further illustrate the invention and with reference to earlier Figures, Figure 5 presents a data flow diagram showing communications 112 between clients 108 and the server 110, and also showing other aspects of the operation of an inventive hub & spoke architecture. At a client entity A, a client A obtains 500 purchase order data from A's single-entity transaction system 106. This may include, for instance, reading purchase order data such as vendor name, part identifier, and desired quantity from a disk or other buffer 114. Client A then embeds 502 the purchase order data in self-documenting form in a communication 112, translating from the system 106 format to the hub-client format using an adapter 204. The embedded purchase order data is an example of electronic commerce transaction data 400. The history 408 is updated. Authentication data 306 and/or error management data 304 are generated if called for by the embodiment. Then client A sends 504 the resulting communication 112 to the hub 110, using the transport adapter 206 and the underlying computer network.

The hub server 110 processes 506 the incoming purchase order 112. In a given situation and a given embodiment, this may include translating a high-level destination ID to a network address; sending a confirmation receipt to a client A; noting a purchase order's arrival time in case the purchase order is not acknowledged by client B quickly enough; and sending data to the ERP module 116. A detailed example of hub processing 506 is illustrated in Figure 10 and discussed below.

As indicated by a step 1000, the server 110 accepts the incoming data through a variety of transport mechanisms, but the eventual target of each of these is a persistent storage area that is easily manageable, such as storage on a hard disk or in a RAID unit with message queues or database tables. In one embodiment, the act of acceptance and conversion of the transmission 112 into a transaction is inherent in the process of placement 1000 into the queue mechanism. Therefore one has a transaction in the queue. In one embodiment, successful addition of the purchase order transaction (for instance) to the queue generates another transaction that is delivered to the client 108 as a transmission 112 to confirm receipt.

During a step 1002 the hub de-envelopes or breaks out the history 408 and routing information 404 embedded 502 in the transaction. The history data 408 is then secured 1004

into a transaction management system to allow for proactive administration of the transaction via the history data. When this is accomplished a history event is added 1004 to the history 408 indicating that the hub server 110 is now managing the transaction life cycle.

During a step 1006 the hub identifies and uses the transaction type and customer 102. One embodiment of the hub 110 uses information retrieved from the transaction data 400 to look up 1006 the proper processing instructions in a central database at the hub 110; in other embodiments the information is stored elsewhere in the signal 300 or the hub 110.

The hub compares 1008 the current step in transaction processing (e.g., "receive purchase order") with the instruction step(s) obtained during step 1006, to identify dependencies, namely, to determine whether some instructions depend on the result of other instructions for their input. If so, the instructions that provide required input are completed before the instructions that depend on the input.

The hub server 110 executes 1010 the defined steps for internal processing of the document. In the case of a purchase order, for instance, the look up 1006 would indicate that the next step in the process after creation is data validation (checking for out-of-range values, verifying checksums and/or digital signatures, etc.) . Data validation is performed during an instance of illustrated step 1010 by executing code which is associated by the hub 110 with the processing step dynamically in response to at least the transaction type.

Once the data is validated the next step is looked up 1006 and executed 1010 in a similar fashion. For example, the next step could be to submit data 400 to the ERP system 116 through a conventional API defined within the ERP system. This would be executed in a synchronous fashion, with the hub 110 awaiting a response from the ERP system 116. The response could be a simple success or failure statement from the ERP system 116 to the hub server 110.

Each step in the process looked up during illustrated step 1006 has a similar executable code association for use with illustrated step 1010. As shown in Figure 10, exception handling code is preferably provided to deal with failed process steps, and this is preferably done regardless of whether the failure is data related or process execution related. The exception code moves 1012 the transaction to a persistent storage designated to hold exception-related transaction and processing information. An administrator or engineer can then examine the information to determine what caused the exception and how to respond further to the exception.

Once successful submission of the purchase order has been achieved the server 110 moves 1014 the transaction to a holding area (queue) that maintains the transaction. The hub 110 then awaits a purchase order acknowledgement, or does whatever is defined as the next step for the document in the transaction processing definition. For example, the hub could await the purchase order acknowledgement from the supplier/vendor 102. Once the acknowledgment 112 is received and processed, the server 110 would move to the next step, and so on. Of course, in particular instances or embodiments, the steps shown in Figure 10 may be reordered, performed concurrently, omitted, renamed, and/or regrouped, in ways that will be apparent to one of skill in the art.

In general, an architecture 100 according to the present invention could be implemented with a hub 110 for which a processing path is defined using a modeling tool like Unified Modeling Language ("UML"). This approach generates new code for each step and stores that code for future use. However, this may be relatively inefficient due to the large code base and duplication of code it suggests.

Accordingly, other embodiments of the system 100 store a step or event in the business process lifecycle (e.g., purchase order lifecycle) as a database entry in a communication 112 that refers to an existing generic piece of code. The hub 110 executes that code with partner, processing instruction step, and process as parameters, as illustrated in Figure 10, for example. This method allows a web based process modeling tool to be built, similar to a mapping website, to define how the system 100 should track the process of executing the purchase order lifecycle. Whereas the UML approach might be marketed with a claim that "no coding is necessary" at a client 102, the latter approach could claim "no systems or servers are necessary" there because the business process is modeled at the central hub 110 and executed at the central hub 110. This process modeling functionality could be distributed to the clients 108 as a "processing model" transaction. This use of references (as opposed to the UML approach) is a preferred implementation of the processing instructions 406 embedded in the document 300. That is, the embedded instructions 406 refer to the executable instructions performed 506/610/612 by the hub 110, allowing one to modify the process model at the client 108 and then activate it by a simple instruction in the transaction signal 300.

Then the hub server 110 sends 508 the purchase order data in a communication 112 to the vendor client entity B. Client B extracts the purchase order data, and submits it to B's system 106. In response to the purchase order, the system 106 generates an invoice. Client B

obtains 510 the invoice data from B's system 106, embeds 512 it in a communication 112 (converting the invoice data format to an XML format used by the hub 110), and sends 514 it to the hub 110. The hub processes 516 the invoice communication as discussed above (another step in the hub processing) and sends 518 the invoice data to client A.

Methods

Figure 6 illustrates methods of the present invention. Steps shown may be omitted, repeated, performed concurrently, reordered, combined, and/or renamed in a particular embodiment, to the extent that the invention continues to operate and also to the extent permitted by the claims. It will also be appreciated that discussion herein of one category of embodiment (e.g., method, architecture, configured storage medium, or signal) of the invention will often be helpful in understanding other categories of embodiment. Unless indicated otherwise to one of skill, therefore, discussions of the methods also pertain to the architecture and signals, discussions of signals also pertain to the methods and architecture, and so on.

During a hub configuring step 600, the hub 110 is initialized according to a definition of the business process model the hub is to implement, including the order in which transactions are to be executed, and the expected steps in each document lifecycle. Also mapping of the documents and their translation could be executed at the hub and stored there for delivery. For instance, mapping defines field relationships so the client 108 can convert data between XML and an internal client 102 format.

During a client 108 installing step 602, client 108 software is installed on computer hardware, the client 108 is connected to the underlying computer network to permit communication with the hub server 110, and the client 108 is given access permissions, connections, and/or other capabilities so it can exchange transaction data with the entity's transaction system 106. Embodiments having pluggable adapters 204, 206, may use a single client 108 installation regardless of formats and/or transports used; the client 108 is configured with downloadable adapters after installation of the data processors 200, 202.

During a related updating step, the client 108 software is updated to include, for instance, more or different adapters 204, 206. The updating step is combined in the illustration with the installing step, but may be viewed as a separate step in other embodiments. Automated client 108 updates are driven by configuration changes, not by an update engine. That is, the client 108 can make changes to itself, or the server 110 can make

changes to the client 108. In either case the changes are reflected to the other system 100 component. This allows the server 110 to be synchronized with client 108 changes, and makes it possible to do mass updates in the system 100 by changing the server 110 configuration and having that change reflected to all clients 108.

From the client entity 102 perspective, the installation process that integrates them in the novel architecture 100 may proceed generally as follows. A new or existing customer 102 chooses to "integrate" by expressing this desire to their sales representative or by choosing the appropriate service on the service provider 104 web site. The entity 102 must exist in the system as a customer before integration can begin. That is, an identifying account that controls access to the web site configuration tools must exist before integration can occur. The integration web site is not a stand-alone site but requires another site to be the homepage and authentication master. A transaction monitoring module in the hub 110 receives an entry from an installation web site or installation software package indicating that this customer 102 has begun integration. As integration commences, the customer 102 is presented with a series of choices and questions. After answering the questionnaire, customer 102 is presented with an installable software package 108. This is another form of a monitored transaction where response time and transactional completeness are "guaranteed" by the monitoring system.

The installation of the client 108 occurs by direct install online, or by a download followed by a local install, using familiar software transfer and installation tools and techniques. During installation the customer 102 is asked more questions about configuration (e.g., host machine, network address to send transactions to and from, name of company, technical contact info, directory or database to look in for transactions). The installation completes and the configuration is sent as a transaction back to the server 110. The client 108 is now installed and the server 110 receives the configuration transaction. An initial test document is sent to the new client 108 automatically; one suitable test document is a catalog with only one item. The client 108 confirms receipt of this test catalog. A transaction monitor in the hub 110 is thus notified that integration installation has completed successfully.

At this point, the transaction monitor is also notified that mapping has started. An email or other message from a service provider 104 integration specialist points the user 102 at an integration web site. Using forms at the web site, the user 102 chooses an output format and document type, and defines output format specifics such as field locations, data elements, and so on. This may be repeated for multiple documents. The user submits a completed

translation map to the hub 110 for testing. The hub 110 generates a sample document using the mapping. The sample document is sent to the client 108 for testing. The transaction monitor is notified that the format translation mapping is being tested. The user 102 returns to the web site, makes changes and repeats the test as need, and ultimately agrees with the document format conversion map(s). The transaction monitor is accordingly notified that mapping has completed for the document type in question. Finally, the document type and customer 102 status are changed to "production, ready for mapped document".

During an obtaining step 604, the client 108 obtains initial transaction data 400 from the system 106. For instance, the client 108 may obtain 500 purchase order data. In some embodiments, the purchase order data is generated in response to choices made by a buyer using a service provider web site 120 and/or a purchasing interface of the client 108. This allows buyers 102 to do some or all of the following: evaluate products from one or more vendors in side-by-side comparisons; search product catalogs by manufacturer, product number, product description, or some combination thereof; create a "favorites" list of products that are ordered often; duplicate previous orders entirely or with specified changes; make personal notes on specific products, to help remember which personnel prefer which product, for example; and access the entity's buying history to identify trends in product usage, for instance.

The website client 120 may be furnished by the same service provider 104 that furnishes the hub 110. However, from the perspective of the hub 110, the website 120 is just another client 102 that submits orders. Indeed, the website client internal architecture includes a client 108, a buffer 14, and some type of order placement software, such as a purchasing interface that includes Internet browser forms and accompanying Java or CGI order placement code. When someone places an order on the web for product that is not included in the system 106 catalog the hub 110 sends a micro-catalog to them before the XML order, to make sure they have the products in their item master in their ERP system.

During an embedding step 606, the client 108 performs any necessary format conversion to a self-documenting format and embeds the reformatted transaction data 400 in a communication 112.

During a sending step 608, the client 108 sends the communication 112 over the computer network "wire" to the hub server 110.

During a processing step 610, the hub server 110 processes the communication 112 by reading the embedded routing information, customer ID and document type, selecting server

stored information about this partner and executing the appropriate business logic. Other processing steps may be executed as deemed necessary by the profile of the partner or as embedded in the transaction. For instance, in the XML file above a line that reads "processInstruct = "/usr/batch/load855.sc"" illustrates the ability of the client 108 to specify a program at the remote location 102 to execute processing instructions as a hub 110 processing step. For convenience and to emphasize its optional nature, a procurement accounting step 612 is shown separately. As discussed earlier, in some embodiments the step 612 and the step 610 are grouped together as a single processing step, such as the step 506. Figure 10 illustrates detailed steps for hub processing 610, 612 in one embodiment; the steps shown in Figure 10 are discussed at least in connection with Figure 5 above.

Steps 614 to 632 proceed in a similar manner. Step 614 sends a communication 112 from the hub 110 to the client 108. For simplicity, a single client B is denoted in Figure 6, but a single purchase order may result in order communications 112 to one or more vendors 102. On receiving the communication, the target client 108 extracts 616 the transaction data from the message 112 and presents 618 the data to the client 102 system 106. If there are additional processing instructions, such as notification of personnel at client B, these are carried out 620. If this is the end of the business lifecycle for the transaction in question, processing ends as indicated by the arrow after step 620. For instance, the communication just processed 620 may have been a confirmation that requires no further communication back to client A.

However, if the communication calls for a response under the business process model, then client B may obtain 622 responsive data (such as an order confirmation number) from its local system 106. Client B embeds 624 that data in an XML or similar format in a responsive communication 112, and sends 626 it to the hub 110. Client 108 interaction with the system 106 through a buffer, format conversion during step 624, and transmission methods using adapters 206, for instance, proceed as discussed elsewhere herein. The hub 110 likewise processes 628 the responsive communication as discussed in connection with illustrated step 506 and elsewhere herein. The hub then sends 630 a corresponding communication 112 back to client A, which extracts 632 it, and then proceeds accordingly in view of the communication 112 content and the business process model that defines the lifecycle for the transaction in question.

Proactive transaction monitoring and exception handling steps 634 are performed as needed through the illustrated steps 600-632. These steps are discussed elsewhere herein in connection with notifications, with timeouts, and with Figure 10, for instance.

A monitoring and exception handling step 634 may be performed, or partially performed, at one or more various points in the sequence just described. In one embodiment, upon receipt of the transaction data 400 it is persistently stored in a location such as a hard disk. This location indicates the transaction 400 arrival time. If the defined response time for a given transactional step expires without a detected response, then the monitoring system in the hub 110 or client 108 reacts by sending out notifications of an expired transaction. This can include automated notification to the integrated system 100, email notification to an account manager either in the partner 102 or the service provider 104. Further steps will be taken if an exception response time expires. For instance, the system 100 may reroute the transaction to an alternate partner 102 or cancel the transaction and provide notification of these events in the same fashion as above.

Implementation Development History

A first implementation of the invention included four classes, and was designed using a traditional procedural design methodology. Functionality characteristics included single threaded message (prototype communication 112) processing, a single output format per message, and adapter 204 support for converting XML to EDI, and converting XML to delimited file output formats.

This implementation included a total of four objects that when instantiated, made a connection to a specified mailbox, and processed mail messages based on configuration settings taken from a properties file. Mail messages were retrieved by the client 108 based on subject heading criteria and were processed in a single threaded mode. Processing involved polling a mail server and retrieving the message 112 body, looking up the processing direction from a properties file, parsing the body and writing the result to a disk file 114. This implementation supported only one processing direction, namely, from XML to EDI or from XML to plain text or a comma delimited format.

A second implementation included eleven classes, and was designed using an object oriented design methodology. Its functionality characteristics included multithreaded mailbox connection, single threaded message 112 processing, multiple output formats per message, properties file delivery to a hub server 110 operated by Promedix.com (assignee of the

present invention), processing response email, a consistent internal order object, adapter 204 support for EDI to XML conversion, and for conversion from XML to a Java order object.

This second implementation included a total of eleven objects that worked together in a more complete object oriented fashion. One managing object in the client 108 controlled other objects that made a connection to a mail server, processed any retrieved messages, and output into resultant formats. Connections to the mail server occurred in child threads forked by the managing object. Within an individual child thread, the message body was retrieved and passed to an object that processed the message into one or many formats indicated by settings in a properties file. These resultant formats included conversions from EDI to XML, and conversions from XML into a Java order object. The Java object could then be converted into a file with any selected delimiter, or any other customized format. In addition to the processing of the message body, a response email with the results of the processing or any exception/error messages could optionally be sent to one or many email addresses. This iteration also monitored and sent its properties file to the Promedix.com hub server 110 for record keeping purposes.

A third implementation included thirty classes, and was designed using methodology directed toward a "black box" application framework design with a publicly available Application Programmers' Interface ("API"). Its functionality characteristics included a "pluggable" (i.e., modular, separable) transport protocol layer for protocol independence through transport adapters 206, a pluggable adapter 204 layer, registration and dynamic update of adapter classes, internal XML configuration and operation semantics, a publicly available API, encryption support, and event logging.

This third implementation represents a fundamental re-design over previous iterations. This implementation's design is a "black box" application framework. Transport layers and adapters can be plugged in (and out) through a publicly available API set, extending the functionality of the software to meet customized needs. The supported transport layer includes client transactions through traditional Simple Mail Transfer Protocol ("SMTP") services; suitable alternatives include adapters 206 for Remote Method Invocation ("RMI") socket use; HyperText Transfer Protocol ("HTTP"), File Transfer Protocol ("FTP"), or other socket-to-socket connections; and other protocols as necessary. Functionality is also extensible through pluggable format adapters 204. These adapters 204 are plugged in and executed in child threads enabling processing to any output format necessary. Output formats

of the adapters 204 support Java Database Connectivity ("JDBC") connections, output into XML or delimited files, and "screen scraping" support to extract data from screen drivers.

In support of the publicly available API set, and the dynamically loaded transport and adapter layers, this implementation executes an internal registry service that monitors and updates classes by sending or "pushing" its current state to the hub server 110. If the hub server identifies a requirement for update, it sends the required class to the client 108 as a response. The required classes are received, registered, and loaded for execution. The registry uses client-specific, dynamically built XML documents from the hub server in its communication with the server to request and load transport and adapter classes that may be new, updated, or installation-specific.

This implementation can also use any encryption standard from a provider that has Java-enabled their encryption algorithm classes. Encryption is used for all incoming and outgoing messages, regardless of transport protocol. Currently synchronous keys are being used, however, asynchronous public-private keys can be generated and used if needed.

Event logging of current execution and exception/error state is also performed in this implementation.

Additional Client and Architecture Examples

Figure 7 illustrates an architecture according to the present invention during use by a transaction community for a market in medical products. A client 108 at a hospital exchanges data through a disk buffer 114 with a transaction system 106. The invention is compatible with conventional approaches, in the sense that it can be used in combination with them. Thus, in addition to the invention the hospital 102 also uses the buffer 114 with a conventional EDI gateway communication system 700 to exchange EDI data directly over conventional value added network 702 with EDI trading partners 704.

With further attention to the invention, the hospital 102 sends and receives communications 112 by email through an otherwise conventional email server 706 that is in turn serviced by a conventional router 708 and protected by a conventional firewall 118. The firewall 118 is prudent because the hospital 102 communicates with other transactional community members 102 in part through the Internet 710, which is generally open to the public but also less expensive and easier to use than many value added networks. The communications 112 travel over the Internet 710 to the hub 110, and communications 112 likewise travel over the Internet 710 between the hub 110 and other client entities, such as

vendors 102 which are not direct EDI vendors but instead use the invention for electronic commerce with the hospital 102.

The illustrated embodiment includes a procurement module 116 at the hub 110. The illustrated procurement module 116 includes a conventional enterprise resource planning system 712 and a conventional warehouse management system 714. Other embodiments may use other software and/or omit one or more of the illustrated systems.

Figure 8 further illustrates data flow within a client 108 embodiment. An incoming data processing flow 800 may be performed with the incoming data processor 200, and an outgoing data processing flow 802 may be performed with the outgoing data processor 202. The email server 706 receives email messages and places them in one or more mailboxes 804 using conventional techniques. The incoming data processor 200 checks 806 the client 108 mailbox 804 for messages. This may be done by background polling, as indicated, by interrupts, or by other familiar techniques. If any email messages are found in the mailbox 804, they are checked 808 to determine if they are electronic commerce communications 112 or other messages pertaining to the invention. If they are communications 112, their transaction data 400 is extracted (as during steps 616, 632, for instance) and formatted for the client entity's internal transaction system 106 (using a format adapter 204, for instance). The transaction data 400 is presented to the system 106 in this case by writing 812 the data 400 to a disk 114. An acknowledgement or other status update is also sent 814 by email to the hub 110, informing the hub 110 of the status of the transaction at the client 108.

During outgoing data processing 802, the outgoing data processor 202 operates in a similar manner. The disk 114 is checked 816 for files that may contain transaction data 400. If it is determined 818 that data 400 has been presented to the client 108 by the client entity's internal transaction system 106, then the client 108 imports 820 the data 400, does format conversion 822 as needed to place the data 400 in an XML or other self-documenting format, embeds the data 400 in a communication 112 and sends 824 the communication 112 to the hub 110 as an email message.

Figure 9 further illustrates steps and structure in a client 108 embodiment. During an initialization step 900, the client 108 reads a registry and loads all installed adapters 204, 206 into memory. During a step 902, the client 108 then reads properties from disk. Properties may specify, for instance, notification addresses, directories 114 to read from, directories114 to write to, and/or configuration information for transport types and listeners. The client 108 also checks to see if adequate disk space is available, and attempts to open an email

connection with the email server 706. As illustrated, if the email connection attempt fails (possibly after retries) then the client 108 notifies 904 a designated administrator or technical contact using a previously specified email address, and preferably also notifies the hub server 110 by alternate email or other means. If the email connection is established but the client 108 lacks necessary disk space or detects an error in the properties, then the client 108 notifies 906 the hub 110 of the error by email.

If no errors are detected, the client 108 enters and repeats a main loop 908. The loop checks 910 for email messages, as discussed for instance in connection with Figure 8. If a communication 112 is received, the illustrated client 108 spawns 912 a separate thread 916 to process the incoming data 400. The main client loop also notifies 914 the hub 110 of status at suitable times. For instance, the hub 110 may be notified 914 that the communication 112 has been received, may be notified 914 that the transaction data 400 within the communication has been processed (by the thread 916), or both.

In the illustrated embodiment, the thread 916 receives 918 the email message as an email object. The thread 916 parses 920 a document object out of the email message. The document object contains the transaction data 400, history, routing, processing instructions, and so on. The thread 916 then parses the document object into a Java object. This facilitates use of a format translation engine based in Java. More generally, the thread or other process places the translation data 400 in memory for format translation by a translation engine into a format usable by the legacy system 106. Finally, the thread 916 writes the formatted data 400 to the buffer 114 to present it to the client entity's local transaction system 106, and sends 926 status back to the main loop 908 so the hub 110 can be notified.

Additional Commercial Document Examples

It will also be appreciated that the invention can be used to advantage with various types of business documents. To further illustrate the inventive architecture, methods/configured media, and signals, several additional examples of the invention in operation are described below. It is not necessary for every embodiment of the present invention to support every document type discussed here, to contain the exact components mentioned, or to perform the precise steps listed. These are merely examples that illustrate possible embodiments.

Inbound Purchase Order 300 from Customer 102 to E-Commerce Service Provider 104

The inbound purchase order from a customer 102 to the hub 110 is a common document that will be implemented in many embodiments. Even the most rudimentary systems 106 support some form of purchase order output. The basic process may be modified, but the following describes a simple flow. The processing of transactions in a preferred embodiment is the same for all business documents and all directions (buyer toward seller or vice versa), in the sense that the processing instructions are abstracted into the hub 110 database and the hub just walks through a process definition table and executes steps accordingly, as discussed, for instance, in connection with Figure 10. The steps outlined below are examples; they could be different for various customers and suppliers:

1. Order created in a materials management ("MM") remote ERP system 106
2. Supplier information is verified against supplier master
3. Validation of order items against item master, including pricing
4. Routing for order defined - potential for external routing mechanism
5. Extract order for external delivery (file, fax, voice, printer, EDI, etc.)
6. Notification or discovery of extracted order
7. Acquisition of file
8. Conversion of file into usable intermediate format based on local mapping data
9. Combination of data with local routing information
10. Combination of data with local mapping data
11. Discovery of routing info
12. Encryption of data, mapping, routing into communication 112
13. Deliver communication 112 to host 110 via identified routing and transport
14. Decrypt communication 112 during receipt of information
15. Receive into queue through transport handler/listener
16. Reply to client with confirmation of receipt
17. Pull transaction from queue
18. Submit order to ERP system through server 110 services

Purchase Order Delivered to Supplier 102 from E-Commerce Service Provider 104

In some embodiments, the inbound purchase order received from a customer into the service provider's system will be entered as a sales order. This sales order will then be converted into a purchase order to be sent out to the supplier 102. The outbound purchase

order is a common document that will be implemented in many embodiments; even the most rudimentary system 106 supports some form of order entry. The basic process may be modified, but the following describes a simple flow.

1. Purchase order created in ERP system 116 (automatic or manual)
2. Supplier information verified against supplier master
3. Validation of order items against item master including pricing
4. Extract order for external delivery (delivery to file system or Java Message Service ("JMS") queue)
5. Notification or discovery of extracted order
6. Acquisition of file/document
7. Discovery of routing info/Routing for order extracted from database
8. Combination of data with routing information
9. Combination of data with mapping data
10. Encryption of data, mapping, routing into communication 112
11. Transaction handed to proper delivery adapter 206
12. Deliver communication 112 to host 110 (hub server) via identified routing and transport
13. Decrypt communication 112 during receipt of info
14. Receive into client 108 at supplier 102 through transport handler/listener
15. Find proper adapter based on document
16. Load mapping info
17. Convert document based upon mapping info
18. Execute adapter output (e.g., write file to disk 114)
19. Reply to host with confirmation of receipt
20. Notify supplier ERP system of transaction arrival - optional

Purchase Order Acknowledgement Submitted by Supplier 102 to E-Commerce Service Provider 104

The inbound acknowledgement of purchase order is a common document that will be implemented in many embodiments. The basic process may be modified, but the following describes a simple flow.

1. Purchase order acknowledgement created in supplier's system (automatic or manual)
2. Routing for order defined - potential for external routing mechanism
3. Extract order for external delivery (file, fax, voice, printer, EDI, etc.)

4. Notification or discovery of extracted order
5. Acquisition of file
6. Conversion of file into usable intermediate format based on local mapping data
7. Combination of data with local routing information
8. Combination of data with local mapping data
9. Discovery of routing info
10. Encryption of data, mapping, routing into communication 112
11. Deliver communication 112 to host 110 via identified routing and transport
12. Decrypt communication 112 during receipt of info
13. Receive into queue through transport handler/listener
14. Reply to client with confirmation of receipt
15. Pull transaction from queue
16. Submit order to ERP system through server 110 services

Purchase Order Acknowledgment Delivered by E-Commerce Service Provider 104 to Customer 102

The outbound acknowledgement of purchase order from service provider 104 to a customer 102 is a common document that will be implemented in many embodiments. This acknowledgement will be sent out as soon as the supplier 102 has sent service provider 104 an acknowledgement of the receipt of the purchase order. The basic process may be modified, but the following describes a simple flow.

1. Purchase order acknowledgement created in ERP system (automatic or manual)
2. Supplier information verified against original purchase order.
3. Validation of item status against original purchase order.
4. Extract order for external delivery (delivery to file system or JMS queue)
5. Notification or discovery of extracted order
6. Acquisition of file/document
7. Discovery of routing info/Routing for order extracted from database
8. Combination of data with routing information
9. Combination of data with mapping data
10. Encryption of data, mapping, routing into communication 112
11. Transaction handed to proper delivery adapter 206
12. Deliver communication 112 to host 110 via identified routing and transport

13. Receive into client 108 through transport handler/listener
14. Decrypt communication 112
15. Find proper adapter based on document
16. Load mapping info
17. Convert document based upon mapping info
18. Execute adapter output (e.g., write file to disk 114)
19. Reply to host 110 with confirmation of receipt
20. Notify supplier ERP system of transaction arrival - optional

Invoice Submitted by Supplier 102

The inbound Invoice is a common document that will be implemented in many embodiments. The basic process may be modified, but the following describes a simple flow.

1. Invoice created in supplier billing system
2. Validation of order items against item master including pricing
3. Routing for invoice defined - potential for external routing mechanism
4. Extract invoice for external delivery (file, fax, voice, printer, EDI, etc..)
5. Notification or discovery of extracted invoice
6. Acquisition of file
7. Conversion of file into usable intermediate format based on local mapping data
8. Combination of data with local routing information
9. Combination of data with local mapping data
10. Discovery of routing info
11. Encryption of data, mapping, routing into communication 112
12. Deliver communication 112 to host 110 via identified routing and transport
13. Decrypt communication 112 during receipt of info
14. Receive into queue through transport handler/listener
15. Reply to client with confirmation of receipt
16. Pull transaction from queue
17. Submit invoice to ERP system through server 110 services

Invoice Delivered to Customer 102

The outbound invoice is a common document that will be implemented in many embodiments. The basic process may be modified, but the following describes a simple flow.

1. Invoice created in ERP system 116 (automatic or manual)
2. Customer information verified against account master
3. Validation of invoice items against item master including pricing
4. Extract invoice for external delivery (delivery to file system or JMS queue)
5. Notification or discovery of extracted invoice
6. Acquisition of file/document
7. Discovery of routing info/Routing for document extracted from database
8. Combination of data with routing information
9. Combination of data with mapping data
10. Encryption of data, mapping, routing into communication 112
11. Transaction handed to proper delivery adapter
12. Deliver communication 112 to host 110 via identified routing and transport
13. Decrypt communication 112 during receipt of info
14. Receive into client through transport handler/listener
15. Find proper adapter based on document
16. Load mapping info
17. Convert document based upon mapping info
18. Execute adapter output (e.g., write file to disk)
19. Reply to host with confirmation of receipt
20. Notify customer ERP system of transaction arrival - optional

Additional Business Processes

The following are some of the additional business processes that may also be implemented in various embodiments:

1. Ability to handle user profiles (create, update, delete)
2. Create/update customer master in ERP along with multiple ship-to and bill-to addresses
3. Create/update supplier master in ERP
4. Ability to create/update/delete catalog items or subset of catalog items based on a product category etc.
5. Ability to handle cXML, CBL etc. based purchase orders
6. Advance shipment notifications
7. Send and update notes (text) per line item in the ERP system
8. Order status inquiry

9. Request for quotation

10. Response to request for quotation

11. Purchase order change request - buyer initiated

12. Purchase order change acknowledgement/request - supplier initiated

End to End Transaction Lifecycle

The following list steps through the various states that a transaction may be in during its life cycle. Each step may represent a full process in an ERP system or a single step in the movement of the transaction.

1. Purchase order ("PO") created by customer on web site

2. Server 110 receives PO through JMS posted message from web site

3. Server 110 acquires message from queue

4. Queue listener hands message to submit order interface

5. Submit order bean(s) sends message into ERP through entity bean layer

6. Entity beans perform system inserts and updates with rollback

7. ERP receives order and validates account and customers

8. Supplier purchase orders generated to fulfill the order

9. Purchase orders extracted from ERP to disk or JMS queue

10. Server 110 picks up outbound PO's and discover routing information

11. Transaction is enveloped to include routing and mapping info

12. Transaction is delivered to the partner (supplier) via transport adapter

13. Individual suppliers receive transactions via transport adapter

14. Supplier's installed client 102 receives transaction and confirm receipt by response

15. Client 102 writes data to disk or database 114

16. Supplier ERP 106 processes order and generates internal response

17. Response to order is written to disk 114

18. Client 108 picks up order and converts to internal format

19. Client 108 encrypts the response and delivers to host 110

20. Host 110 decrypts the transaction and puts in inbound queue

21. Server picks up queue entry and process line status into ERP system

22. Generates notification to customer of change in ship status of order

23. Customers client writes PO update information to disk and sends notification to host

24. Customers ERP system picks up order updates and changes line item status

25. When ERP 116 runs invoice batch, exports invoice for this order
26. Invoice is sent with others through outbound queue
27. Invoice received at customer client and written to disk
28. Notification of delivery is returned to server 110
29. Customer ERP imports invoice and enters into AP
30. Payment is generated by customer ERP 106
31. Payment notification is written to disk
32. Client picks up payment notification and sends to host 110
33. Payment notice is written to ERP system 116

Summary

In summary, the present invention provides methods, systems, and configured storage media for electronic commerce. In a hub & spoke architecture such as the illustrated architecture 100, communications 112 are exchanged using XML or another self-documenting format. The communications 112 include electronic commerce transaction data 400 such as purchase order data, processing instructions (which may refer to additional instructions stored at the hub), and a substantially complete history 408 of the transaction's life thus far within the architecture. By keeping the history 408 within the communications 112 that travel around the architecture 100 instead of keeping separate histories or logs at the hub and at the client spokes, the invention permits more efficient and effective proactive transaction monitoring.

Embodiments of the invention also provide other advantages. For instance, by allowing processing instructions to be maintained at the hub 110 and be merely referenced in the communications 112, the invention allows the processing of documents 112 to be tailored to the individual circumstances of each document without using unnecessary bandwidth or risking inconsistency between copies of the processing code. By utilizing pluggable adapters 204 for format conversion between XML and EDI or other proprietary client formats, and pluggable adapters 206 for transport mechanisms, the invention makes the process of adding a client spoke to the hub & spoke architecture more flexible and faster than otherwise.

Unlike conventional systems that simply use the Internet to transport otherwise conventional EDI messages, the present invention does not require that each client 108 know detailed information about each other client 108 for commercial communications 112 to be routed; the hub 110 maintains such information. Indeed, in a given embodiment, the hub 110

may select vendors 102 in cases where the client 102 placing a purchase order has specified items and item counts but has not expressly specified or limited the choice of vendor.

Articles of manufacture within the scope of the present invention include a computer-readable storage medium in combination with the specific physical configuration of a substrate of the computer-readable storage medium. The substrate configuration represents data and instructions which cause the computers to operate in a specific and predefined manner as described herein. Suitable storage devices include hard disks, CD-ROMs, and other media readable by computers. Each such medium tangibly embodies a program, functions, and/or instructions that are executable by the computer system to perform steps substantially as described herein to facilitate electronic commerce.

Although particular methods embodying the present invention are expressly illustrated and described herein, it will be appreciated that system and configured storage medium embodiments may also be formed according to the methods of the present invention. Unless otherwise expressly indicated, the descriptions herein of methods of the present invention therefore extend to corresponding systems and configured storage media, and the descriptions of systems and configured storage media of the present invention extend likewise to corresponding methods.

In addition, the method steps discussed may be performed in various orders, except in those cases in which the results of one step are required as input to another step. Likewise, steps may be omitted unless called for in issued claims, regardless of whether they are expressly described as optional in this Detailed Description. Steps may also be repeated, or combined, or named differently.

As used herein, terms such as "a" and "the" and item designations such as "client" and "communication" are generally inclusive of one or more of the indicated item. In particular, in the claims a reference to an item normally means at least one such item is required.

The invention may be embodied in other specific forms without departing from its essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. Headings are for convenience only. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed and desired to be secured by patent is:

CLAIMS

1. A hub & spoke architecture for electronic commerce comprising a hub server and at least two clients, each client being a spoke of the hub & spoke architecture, each client residing at and acting on behalf of a different client entity, the hub server maintained by a third entity which is an electronic commerce service provider, wherein:

the hub server comprises:

at least one connection whereby the hub server receives communications from the client entities and sends communications to the client entities, the communications from the hub server being sent in accordance with the respective business roles of the client entities, the communications comprising encapsulated process information;

the encapsulated process information in at least one of the communications comprising electronic commerce transaction data in a self-documenting format and also comprising at least one of (a) a substantially complete history of that electronic commerce transaction data within the hub & spoke architecture, (b) a processing instruction reference identifying code at the hub server for executing instructions to process that electronic commerce transaction data, and (c) a specification of at least one item which does not specify a vendor for the item, the hub server instead selecting a vendor for the item, the vendor being within the hub & spoke architecture;

and

each client comprises:

an incoming data processor which checks for incoming communications from the hub server, extracts electronic commerce transaction data from such communications, and posts the data to a single-entity-transaction-system; and

an outgoing data processor which checks for outgoing electronic commerce transaction data from the single-entity-transaction-system, embeds at least part of the electronic commerce transaction data in a communication in a self-documenting format, and sends the communication to the hub server.

2. The electronic commerce architecture of claim 1, wherein at least one client resides behind a firewall at that client's entity.

3. The electronic commerce architecture of claim 1, wherein the electronic commerce transaction data in at least one communication uses extensible markup language to provide a self-documenting format.

4. The electronic commerce architecture of claim 1, wherein the hub server and the clients send communications to one another by email.

5. The electronic commerce architecture of claim 1, wherein the hub server and the clients send communications to one another using message-based middleware.

6. The electronic commerce architecture of claim 1, wherein the hub server and the clients send communications to one another using a socket-to-socket connection.

7. The electronic commerce architecture of claim 1, wherein the hub server receives communications by way of an inbound message queue and sends communications by way of an outbound message queue.

8. The electronic commerce architecture of claim 1, wherein the incoming data processor posts the electronic commerce transaction data by writing it to nonvolatile storage which is also accessed by the single-entity-transaction-system.

9. The electronic commerce architecture of claim 1, wherein at least one client comprises a dynamically loaded transport layer which facilitates transport of communications between the client and the hub server.

10. The electronic commerce architecture of claim 1, wherein at least one client comprises a dynamically loaded format adapter which converts electronic commerce transaction data between the self-documenting format and an electronic data interchange format.

11. The electronic commerce architecture of claim 1, wherein the hub server comprises a database of code for executing instructions to process a communication containing electronic commerce transaction data, and the communication includes a processing instruction reference identifying such code.

12. The electronic commerce architecture of claim 1, wherein hub server code for executing instructions to process a communication containing electronic commerce transaction data accepts a parameter identifying a trading partner.

13. An electronic commerce method for producing a signal embodied in a distributed computing system, the method comprising:

including in the signal electronic commerce transaction data in a self-documenting format;

including in the signal a transaction type which identifies the type of commercial transaction to which the electronic commerce transaction data pertains;

including in the signal routing information for routing the signal to at least one of a client and a hub server; and

including in the signal at least one of:

a substantially complete history which lists at least one event involving the electronic commerce transaction data, the history listing at least an event which generated the electronic commerce transaction data;

electronic commerce transaction data which specifies at least one item without specifying a vendor for the item because the hub server will select a vendor for that item; and

a processing instruction reference identifying code in a database of code at the hub server for executing instructions to process the electronic commerce transaction data.

14. The method of claim 13, wherein the method includes in the signal electronic commerce transaction data in a self-documenting format which includes extensible markup language.

15. The method of claim 13, wherein the transaction type identifies the commercial nature of the electronic commerce transaction data as a purchase order.

16. The method of claim 13, wherein the transaction type identifies the commercial nature of the electronic commerce transaction data as an invoice.

17. The method of claim 13, wherein the transaction type identifies the commercial nature of the electronic commerce transaction data as a payment.

18. The method of claim 13, wherein the transaction type identifies the commercial nature of the electronic commerce transaction data as a catalog.

19. The method of claim 13, wherein the routing information also specifies a transport mechanism for transporting the electronic commerce transaction data.

20. The method of claim 13, wherein the routing information also specifies a format adapter for converting the format of the electronic commerce transaction data.

21. The method of claim 13, wherein the method includes in the signal a processing instruction reference to a database of code outside the signal for executing instructions to process the electronic commerce transaction data.

22. A method for integrating disparate systems into transaction communities to facilitate electronic commerce, the disparate systems acting on behalf of distinct client entities, a transaction community including a plurality of client entities which the method relates to one another during a commercial transaction according to their business roles in the transaction, the method comprising the steps of:

obtaining at a first client purchase order data for an electronic commerce transaction, the first client being in communication with a first single-entity-transaction-system;

embedding the purchase order data in a first communication which uses extensible markup language to document the purchase order data, the first communication also containing at least one of (a) a history of the purchase order data which includes an entry indicating that the embedded data is purchase order data and that the embedded data was obtained at the first client, (b) a specification of at least one item which does not specify a vendor for the item, and (c) a processing instruction reference identifying code in a database of code at an electronic commerce hub server for executing instructions to process the purchase order;

sending the first communication from the first client to an electronic commerce hub server;

selecting a vendor for any item which has no vendor specified in the first communication; and

sending a second communication from the hub server to a second client, the second client being in communication with a second single-entity-transaction-system, the second communication using extensible markup language to document the purchase order data.

23. The method of claim 22, wherein the obtaining step is preceded by the steps of installing the first client and installing the second client, each client is installed at a different client entity, and the hub server is maintained by a third entity which is an electronic commerce service provider.

24. The method of claim 23, wherein the method further comprises the step of remotely updating at least one of the clients.

25. The method of claim 22, wherein the method further comprises the step of confirming receipt of at least one of the communications.

26. The method of claim 22, wherein the method further comprises the steps of:

obtaining at the second client invoice data corresponding to at least a portion of the purchase order data;

embedding the invoice data in a third communication which uses extensible markup language to document the invoice data;

sending the third communication from the second client to the hub server; and

sending a fourth communication from the hub server to the first client, the fourth communication using extensible markup language to document the invoice data.

27. The method of claim 22, wherein the embedding step comprises converting the purchase order data from an electronic data interchange format into an extensible markup language format.

28. The method of claim 22, wherein the embedding step comprises reading the purchase order data from a database and placing the purchase order data in an extensible markup language format.

29. The method of claim 22, wherein at least one of the sending steps comprises sending encrypted data.

30. The method of claim 22, wherein the step of sending a second communication from the hub server to a second client is followed by the step of sending to at least one person identified in the second communication an email notification that the second communication has arrived.

31. The method of claim 22, wherein a procurement accounting step is at least started between the step of sending the first communication from the first client to a hub server and the step of sending a second communication from the hub server to a second client, and the procurement accounting step submits the purchase order data to procurement accounting software on the hub server.

32. The method of claim 22, wherein the method comprises proactive transaction monitoring to determine whether an expected communication has arrived at the hub server and to take an exception-handling step in the event that an expected communication has not arrived.

33. The method of claim 22, wherein the method comprises use of a pluggable transport adapter.

34. A computer-readable storage medium configured with software for performing electronic commerce method steps according to any one or more of the methods of claims 13 through 33.

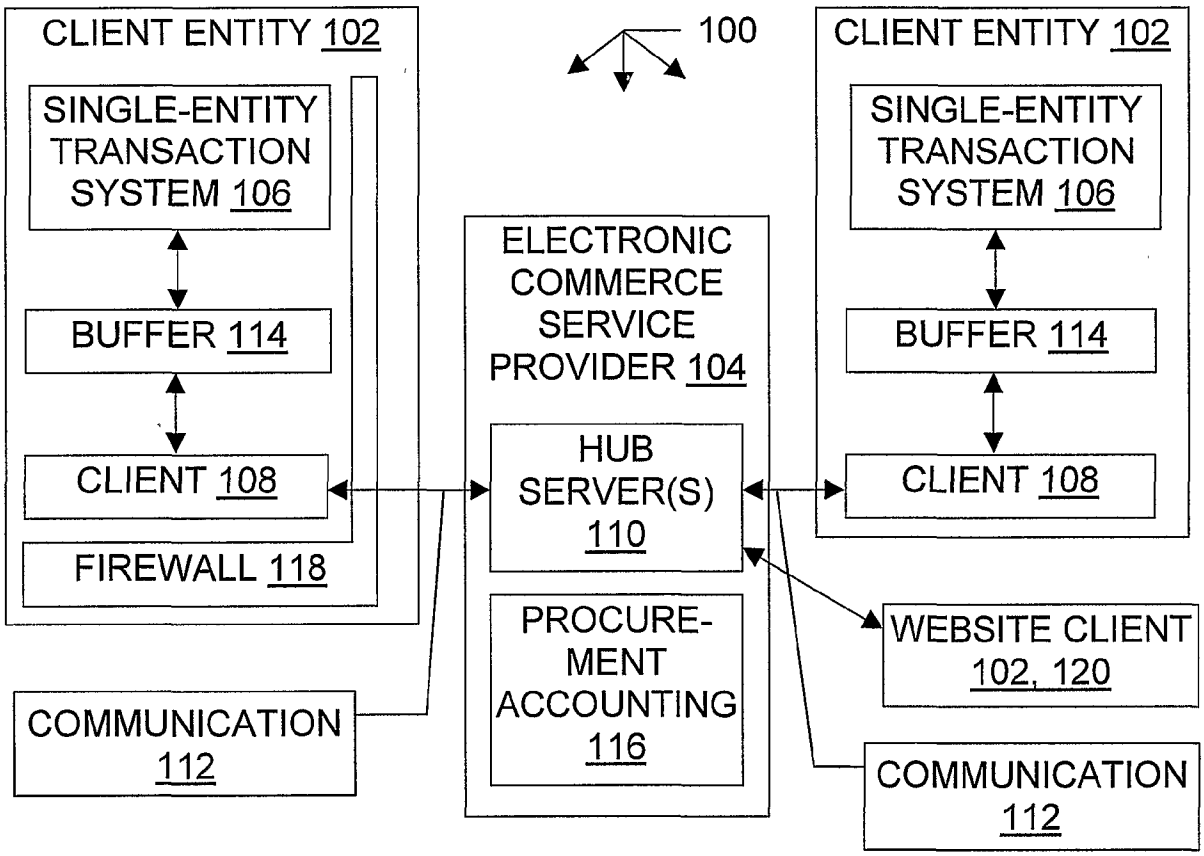


Fig. 1

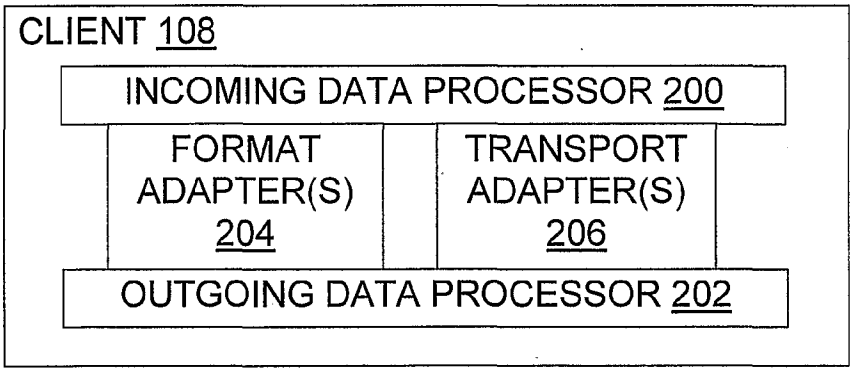


Fig. 2

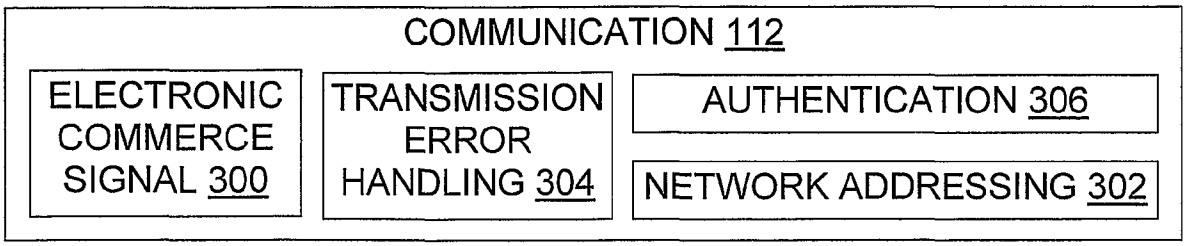


Fig. 3

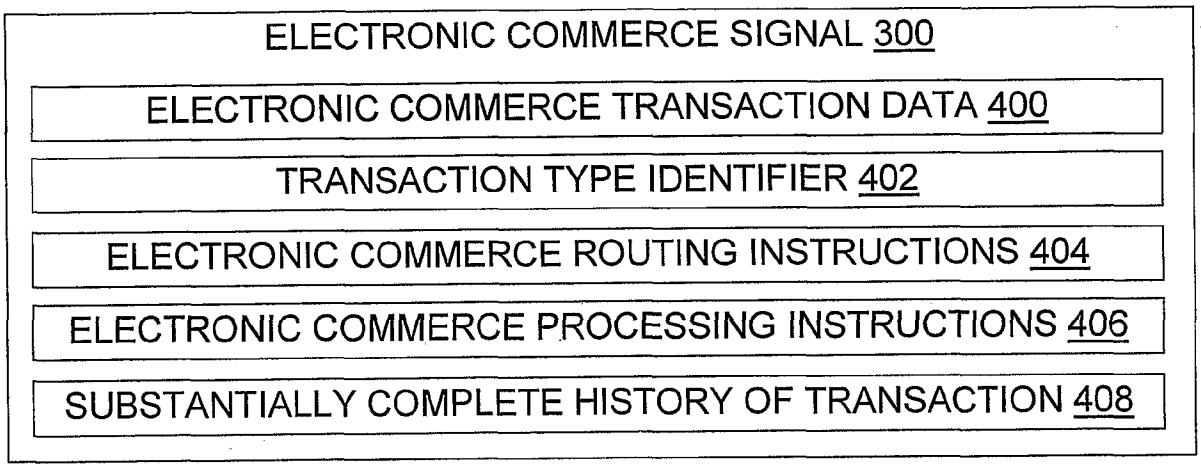

ELECTRONIC COMMERCE SIGNAL 300
ELECTRONIC COMMERCE TRANSACTION DATA 400
TRANSACTION TYPE IDENTIFIER 402
ELECTRONIC COMMERCE ROUTING INSTRUCTIONS 404
ELECTRONIC COMMERCE PROCESSING INSTRUCTIONS 406
SUBSTANTIALLY COMPLETE HISTORY OF TRANSACTION 408

Fig. 4

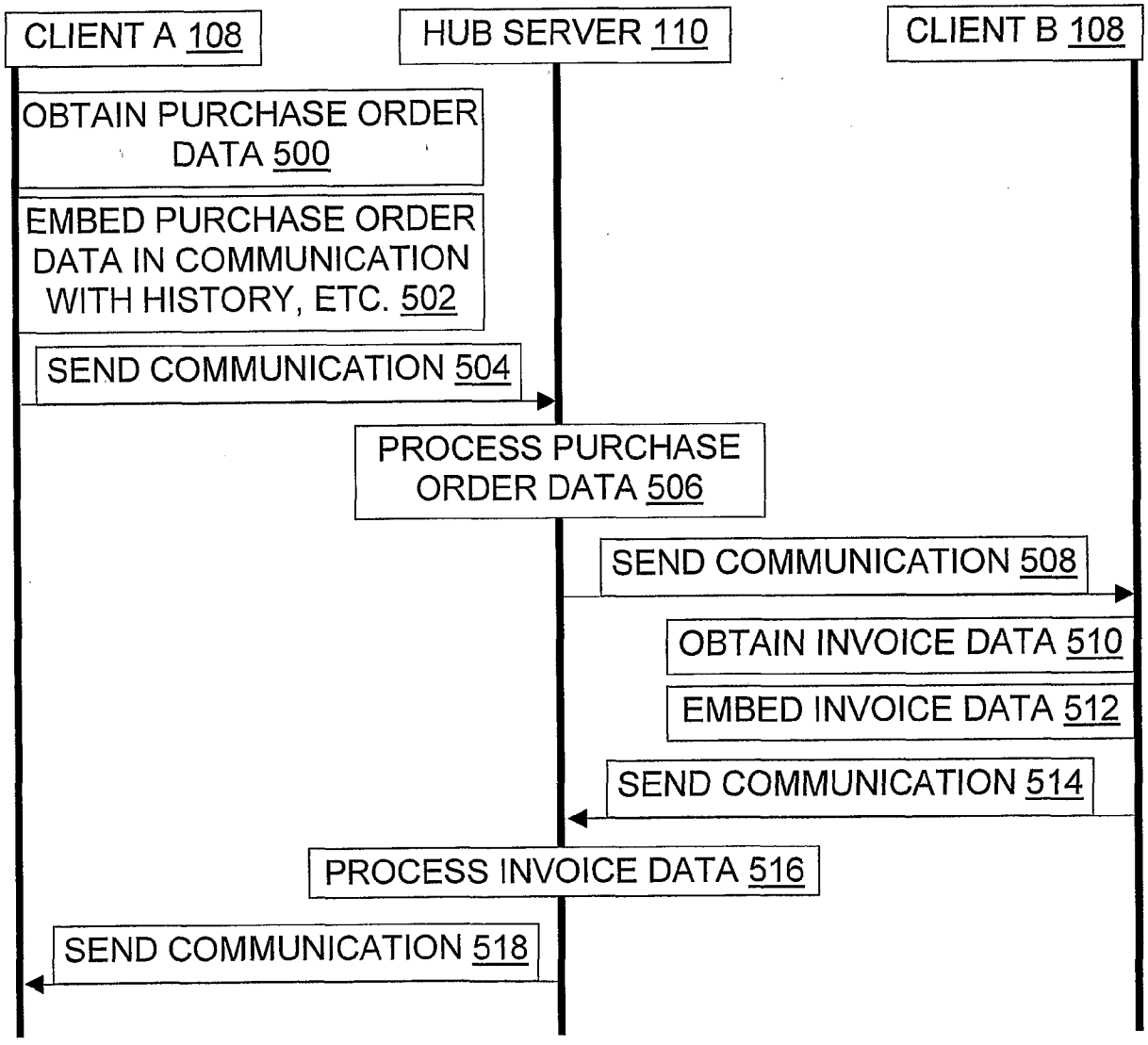

CLIENT A 108
HUB SERVER 110
CLIENT B 108
OBTAIN PURCHASE ORDER DATA 500
EMBED PURCHASE ORDER DATA IN COMMUNICATION WITH HISTORY, ETC. 502
SEND COMMUNICATION 504
PROCESS PURCHASE ORDER DATA 506
SEND COMMUNICATION 508
OBTAIN INVOICE DATA 510
EMBED INVOICE DATA 512
SEND COMMUNICATION 514
PROCESS INVOICE DATA 516
SEND COMMUNICATION 518

Fig. 5

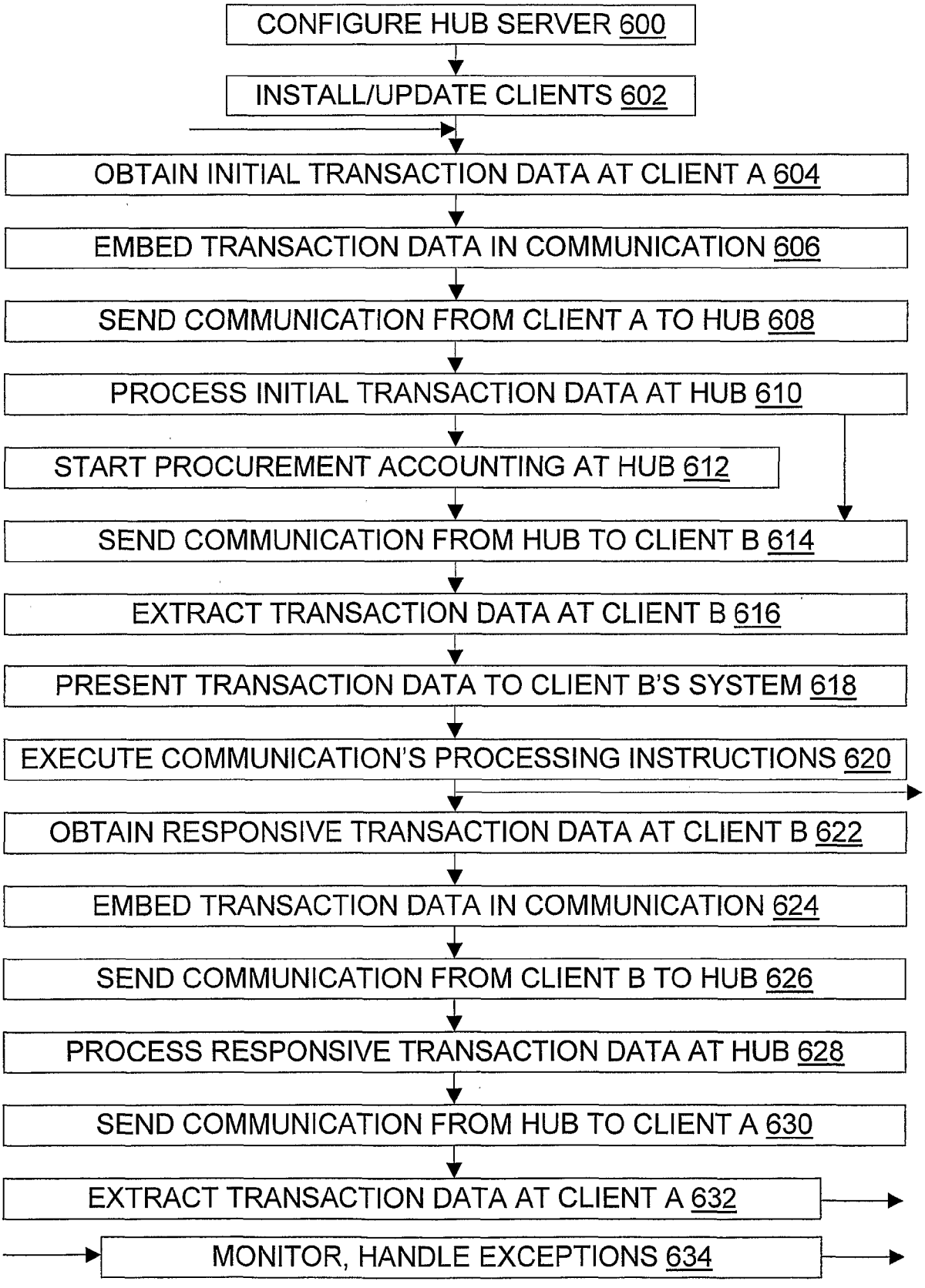

CONFIGURE HUB SERVER 600
INSTALL/UPDATE CLIENTS 602
OBTAIN INITIAL TRANSACTION DATA AT CLIENT A 604
EMBED TRANSACTION DATA IN COMMUNICATION 606
SEND COMMUNICATION FROM CLIENT A TO HUB 608
PROCESS INITIAL TRANSACTION DATA AT HUB 610
START PROCUREMENT ACCOUNTING AT HUB 612
SEND COMMUNICATION FROM HUB TO CLIENT B 614
EXTRACT TRANSACTION DATA AT CLIENT B 616
PRESENT TRANSACTION DATA TO CLIENT B'S SYSTEM 618
EXECUTE COMMUNICATION'S PROCESSING INSTRUCTIONS 620
OBTAIN RESPONSIVE TRANSACTION DATA AT CLIENT B 622
EMBED TRANSACTION DATA IN COMMUNICATION 624
SEND COMMUNICATION FROM CLIENT B TO HUB 626
PROCESS RESPONSIVE TRANSACTION DATA AT HUB 628
SEND COMMUNICATION FROM HUB TO CLIENT A 630
EXTRACT TRANSACTION DATA AT CLIENT A 632
MONITOR, HANDLE EXCEPTIONS 634

Fig. 6

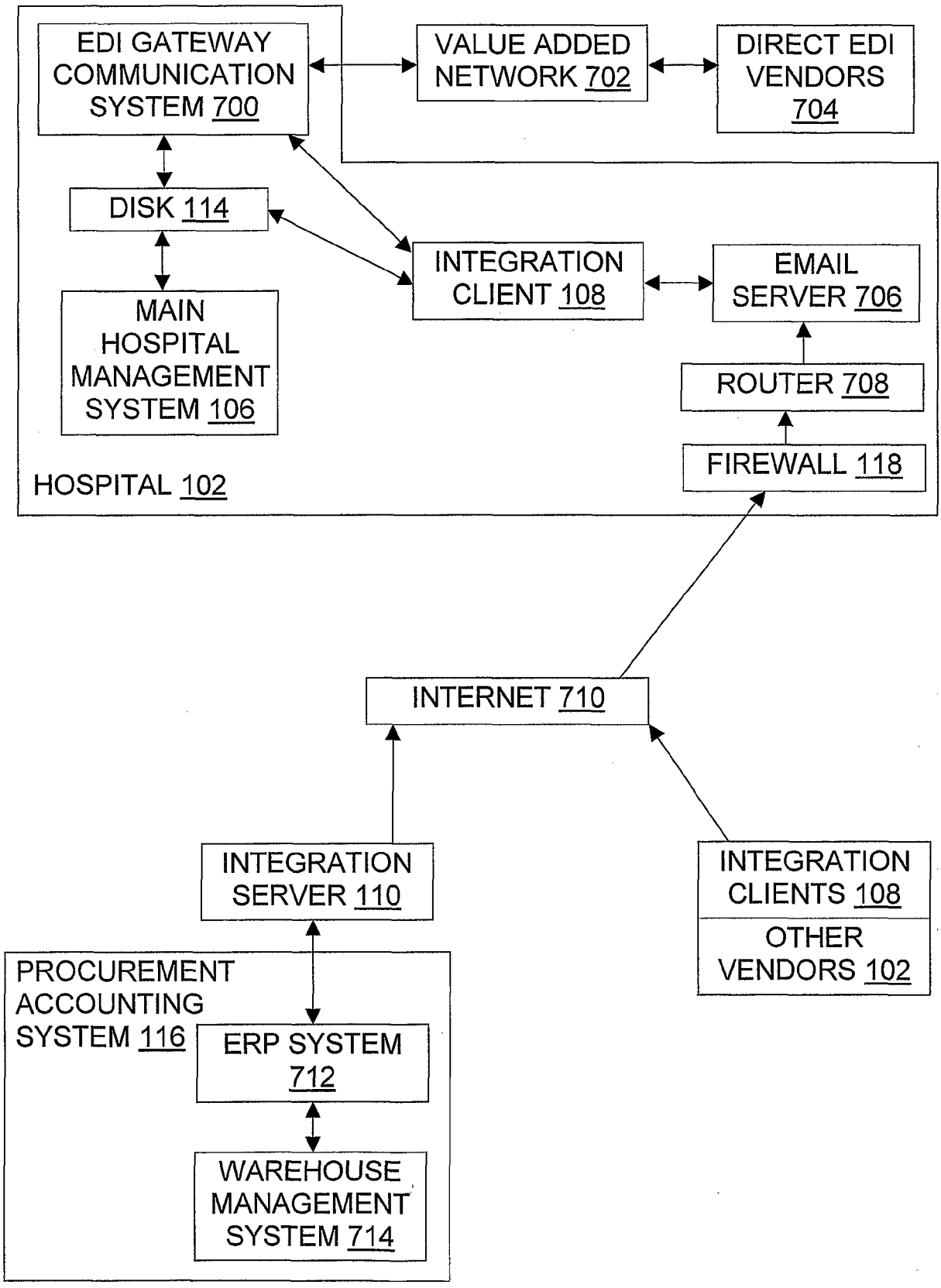

EDI GATEWAY COMMUNICATION SYSTEM 700
VALUE ADDED NETWORK 702
DIRECT EDI VENDORS 704
DISK 114
INTEGRATION CLIENT 108
EMAIL SERVER 706
MAIN HOSPITAL MANAGEMENT SYSTEM 106
ROUTER 708
FIREWALL 118
HOSPITAL 102
INTERNET 710
INTEGRATION SERVER 110
INTEGRATION CLIENTS 108
OTHER VENDORS 102
PROCUREMENT ACCOUNTING SYSTEM 116
ERP SYSTEM 712
WAREHOUSE MANAGEMENT SYSTEM 714

Fig. 7

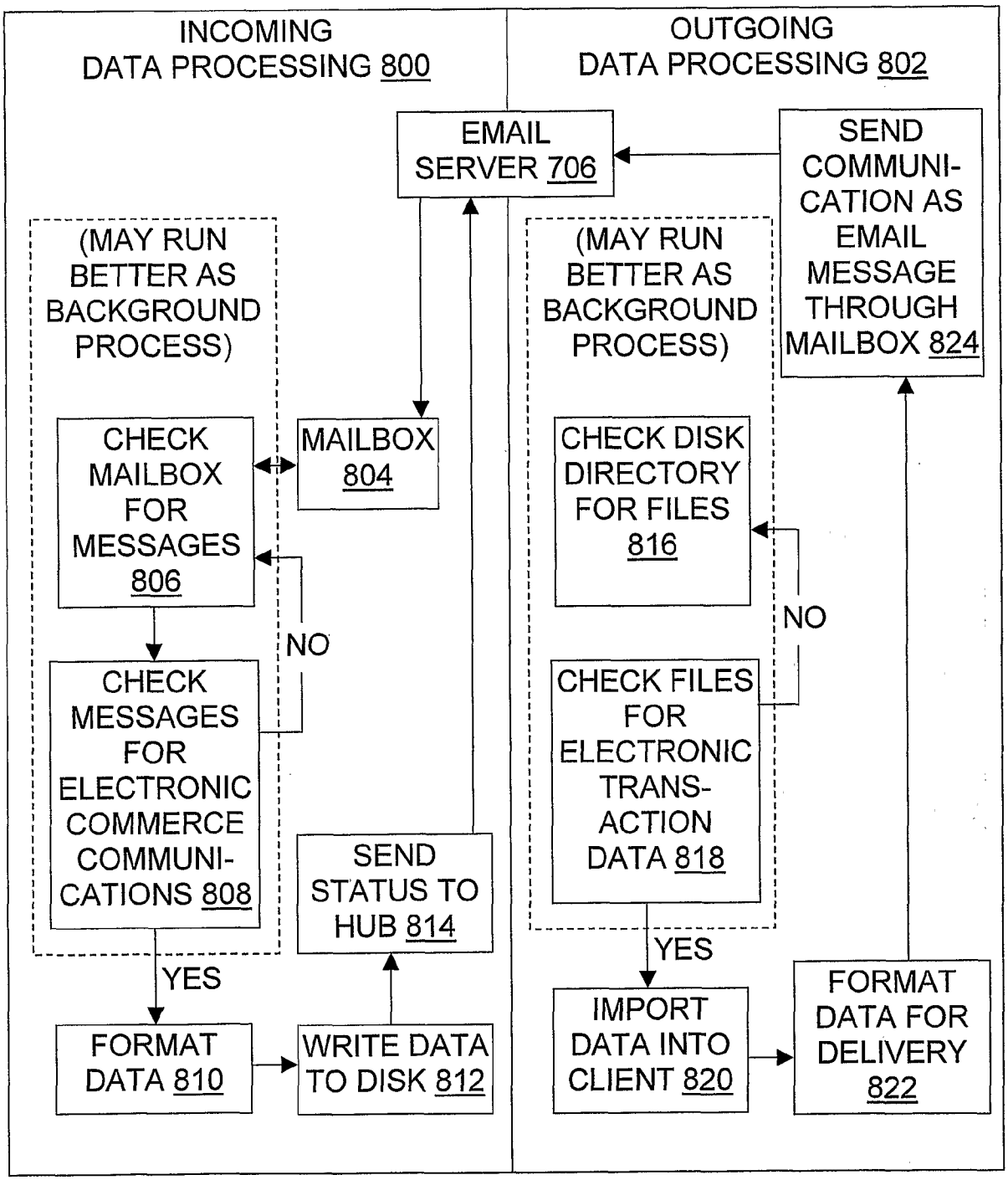

INCOMING DATA PROCESSING 800
OUTGOING DATA PROCESSING 802
EMAIL SERVER 706
SEND COMMUNI-CATION AS EMAIL MESSAGE THROUGH MAILBOX 824
(MAY RUN BETTER AS BACKGROUND PROCESS)
(MAY RUN BETTER AS BACKGROUND PROCESS)
CHECK MAILBOX FOR MESSAGES 806
MAILBOX 804
CHECK DISK DIRECTORY FOR FILES 816
NO
NO
CHECK MESSAGES FOR ELECTRONIC COMMERCE COMMUNI-CATIONS 808
CHECK FILES FOR ELECTRONIC TRANS-ACTION DATA 818
SEND STATUS TO HUB 814
YES
YES
FORMAT DATA 810
WRITE DATA TO DISK 812
IMPORT DATA INTO CLIENT 820
FORMAT DATA FOR DELIVERY 822

Fig. 8

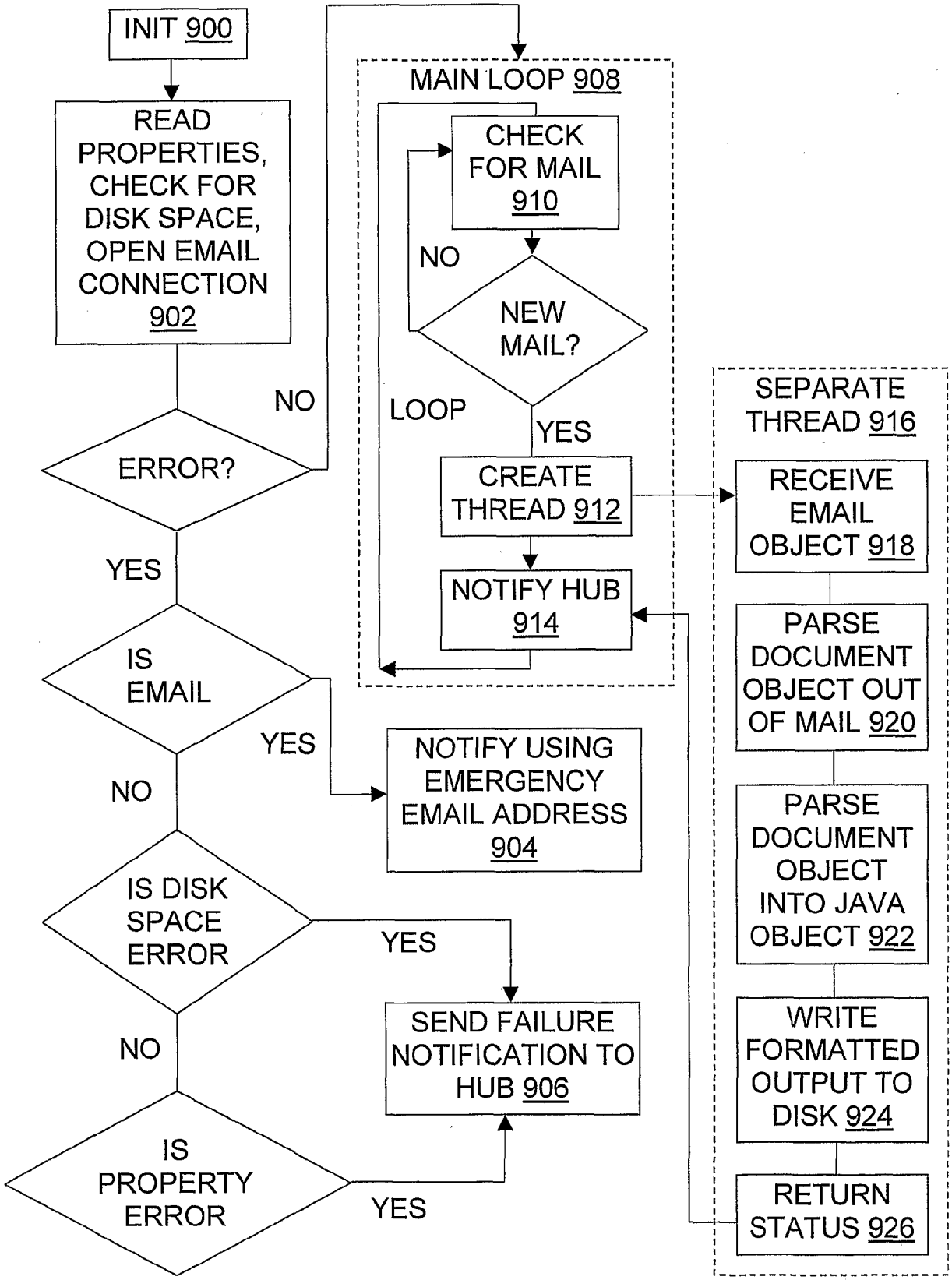

INIT 900
READ PROPERTIES, CHECK FOR DISK SPACE, OPEN EMAIL CONNECTION 902
ERROR?
NO
YES
IS EMAIL
YES
NO
NOTIFY USING EMERGENCY EMAIL ADDRESS 904
IS DISK SPACE ERROR
YES
NO
SEND FAILURE NOTIFICATION TO HUB 906
IS PROPERTY ERROR
YES
MAIN LOOP 908
CHECK FOR MAIL 910
NO
NEW MAIL?
LOOP
YES
CREATE THREAD 912
NOTIFY HUB 914
SEPARATE THREAD 916
RECEIVE EMAIL OBJECT 918
PARSE DOCUMENT OBJECT OUT OF MAIL 920
PARSE DOCUMENT OBJECT INTO JAVA OBJECT 922
WRITE FORMATTED OUTPUT TO DISK 924
RETURN STATUS 926

Fig. 9

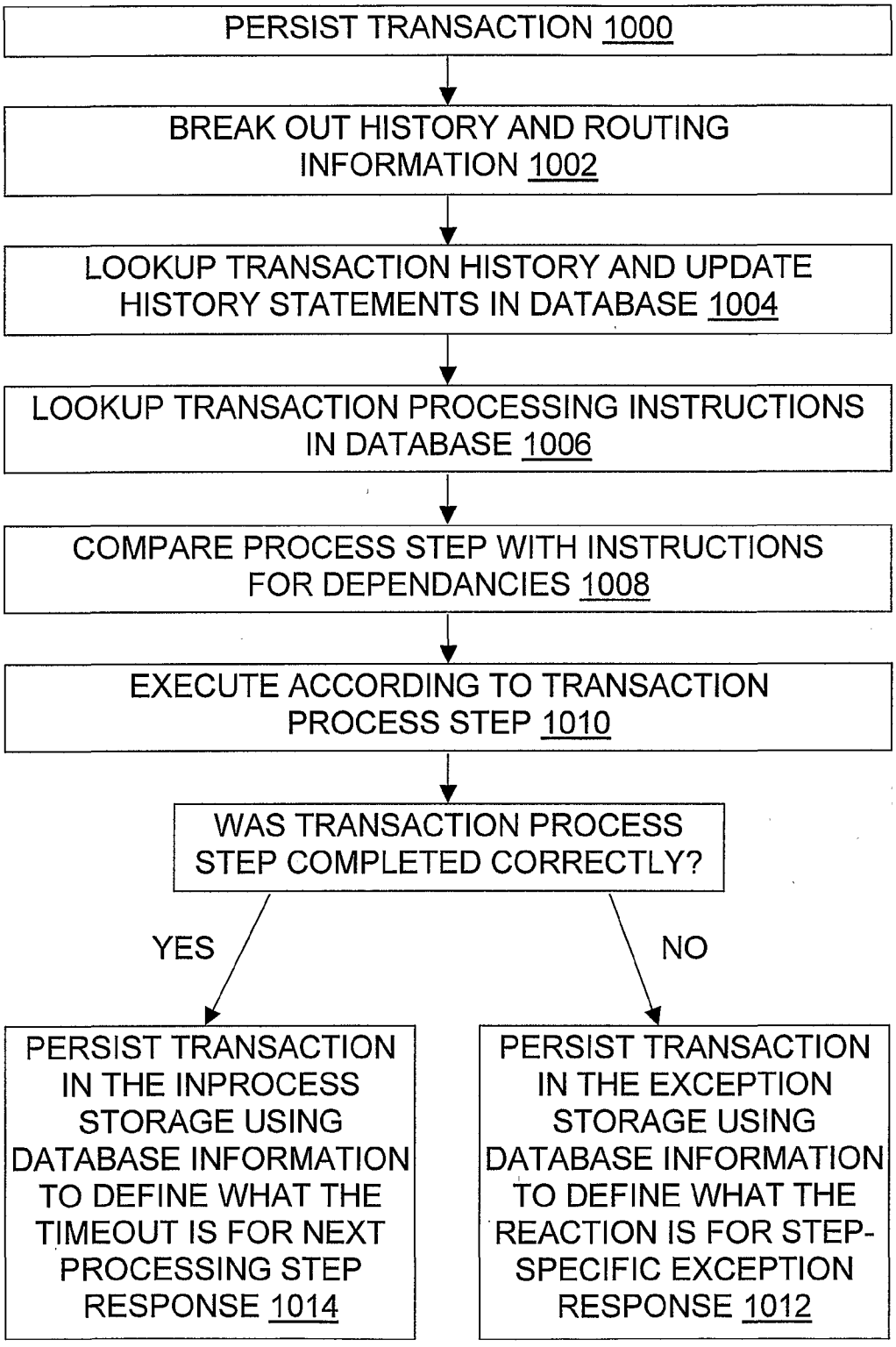

PERSIST TRANSACTION 1000
BREAK OUT HISTORY AND ROUTING INFORMATION 1002
LOOKUP TRANSACTION HISTORY AND UPDATE HISTORY STATEMENTS IN DATABASE 1004
LOOKUP TRANSACTION PROCESSING INSTRUCTIONS IN DATABASE 1006
COMPARE PROCESS STEP WITH INSTRUCTIONS FOR DEPENDANCIES 1008
EXECUTE ACCORDING TO TRANSACTION PROCESS STEP 1010
WAS TRANSACTION PROCESS STEP COMPLETED CORRECTLY?
YES
NO
PERSIST TRANSACTION IN THE INPROCESS STORAGE USING DATABASE INFORMATION TO DEFINE WHAT THE TIMEOUT IS FOR NEXT PROCESSING STEP RESPONSE 1014
PERSIST TRANSACTION IN THE EXCEPTION STORAGE USING DATABASE INFORMATION TO DEFINE WHAT THE REACTION IS FOR STEP-SPECIFIC EXCEPTION RESPONSE 1012

Fig. 10

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 August 2001 (09.08.2001)

PCT

(10) International Publication Number
WO 01/57613 A3

(51) International Patent Classification[7]: G06F 17/60

(74) Agent: OGILVIE, John, W., L.; Computer Law++, 1211 East Yale Avenue, Salt Lake City, UT 84105 (US).

(21) International Application Number: PCT/US01/03087

(22) International Filing Date: 31 January 2001 (31.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/496,361 1 February 2000 (01.02.2000) US

(71) Applicant: VENTRO CORPORATION [US/US]; 1500 Plymouth Street, Mountain View, CA 94043 (US).

(72) Inventors: MACKAY, Robert, R.; 403 East Cattail Court, Saratoga Springs, UT 84043 (US). TRUJILLO, Kristopher, O.; 576 North 2310 West, Provo, UT 84601 (US). BLOYER, Donald, R.; 1054 Lincoln Lane, Park City, UT 84098 (US). BERTOLA, Benjamin, K.; 2085 Pinnacle Terrace Way #306, Salt Lake City, UT 84121 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

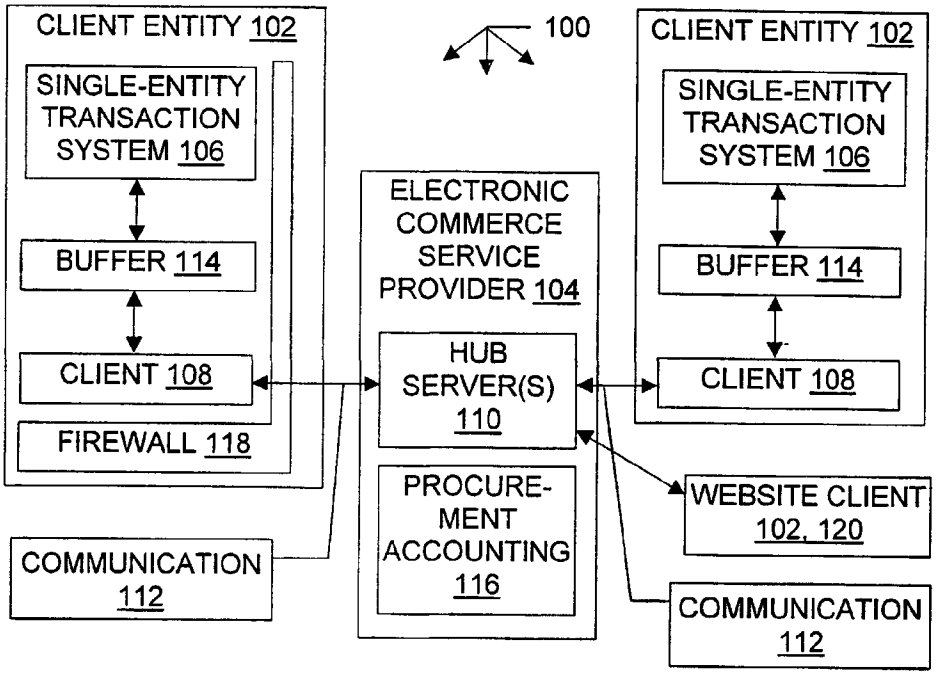
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
11 April 2002

[Continued on next page]

(54) Title: HUB & SPOKE ARCHITECTURE AND METHODS FOR ELECTRONIC COMMERCE



(57) Abstract: Business-business electronic commerce using standardized formats including XML and EDI. ERP interfaces. Hub and spoke architecture (100) allows exchange of purchase orders, invoices, other types of communications (112). Relevant transaction history is sent as part of messages. Client-side software includes firewall, interfaces to accounting and inventory systems.

WO 01/57613 A3

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/03087

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/60
US CL : 705/26

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 705/26

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Dialog

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 6,141,653A (CONKLIN et al.) 31 Oct. 2000 (31.10.2000), column1, line 1 through column 13, line 63; column 10, lines 5-10; column 14, lines 1-67; column 20, line 35 through column 21, line 38; column 22, lines 8-29; column 25, lines 1-67	1, 13, 22, 34 ---------- 2-12, 14-21, 23-33
Y	US 6,125,391 A (MELTZER et al.), 26 September 2000 (26.09.2000), column 1, lines 38 through column 21, line 24; column 8, line 21 through column 12, line 21; column 83, lines 45-67	2-12, 14-21, 23-33, 34

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 November 2001 (30.11.2001)	11 JAN 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer James Zurita Peggy Harwood Telephone No. 703-605-4966

Form PCT/ISA/210 (second sheet) (July 1998)